Mäandrierend durch ein Klima-Wirrwarr

geschrieben von Chris Frey | 20. Februar 2022

Willis Eschenbach

Ein Grund, warum ich immer zögere, über die Motive anderer Leute zu spekulieren, ist, dass ich die Hälfte der Zeit keine Ahnung von meinen eigenen Motiven habe.

Also … aus den üblichen unbekannten Gründen und mit den üblichen unbekannten Motiven begann ich über das GISS-Klimamodell nachzudenken, das als „GISS GCM ModelE" bekannt ist, oder wie ich es nenne, das „MuddleE".

Wie viele solcher Klimamodelle wurde es nicht vor der Konstruktion entworfen. Stattdessen ist es über Jahrzehnte hinweg durch Hinzufügen neuer Teile, Weisheiten zur Behebung von Problemen, Ad-hoc-Änderungen zur Lösung neuer Probleme und dergleichen gewachsen. Oder um einen der Hauptprogrammierer von GISS, Gavin Schmidt, in einer Studie zu zitieren, in dem das ModelE beschrieben wird:

Die Entwicklung eines GCM ist ein ständiger Prozess kleinerer Ergänzungen und Korrekturen in Verbindung mit dem gelegentlichen Austausch bestimmter Teile.

Ein zusätzlicher Schwierigkeitsfaktor ist, wie bei vielen Programmen dieser Art, dass es in der Computersprache FORTRAN geschrieben ist … die 1983, als MuddleE geboren wurde, eine ausgezeichnete Wahl war, aber für 2022 eine schreckliche Sprache ist.

Wie sehr ist es gewachsen? Nun, ohne die Header-Dateien und Include-Dateien und so weiter, nur der FORTRAN-Code selbst, hat 441.668 Zeilen Code … und er kann nur auf einem Supercomputer wie diesem laufen:

Abbildung: Der Cray ecoplex NOAA GAEA Supercomputer, der für die Modellierung verwendet wird. Gaea wurde durch eine Investition in Höhe von 73 Millionen Dollar im Rahmen des American Reinvestment and Recovery Act von 2009 durch eine gemeinsame Partnerschaft zwischen der NOAA und dem Energieministerium finanziert.

Also nahm ich mir das GISS-Modell vor, um zu sehen, was ich finden kann. Von einer Reise, die ich vor zwei Jahrzehnten durch den MuddleE-Code unternommen hatte, wusste ich, dass es Probleme mit „Schmelzbecken“ gab. Das sind die Schmelzwasserbecken, die sich auf dem saisonalen Meereis bilden. Sie sind wichtig für die Berechnung der Albedo des Meereises. Auf meiner letzten Reise hatte ich festgestellt, dass die Tage, an denen sich Schmelztümpel bilden konnten, zeitlich stark begrenzt waren.

Dies führt mich zu einem sehr wichtigen Thema – der erstaunlichen Stabilität des Klimasystems. Es stellt sich heraus, dass moderne Klimamodelle nur schwer auf Kurs bleiben können. Es handelt sich um „iterative“ Modelle, was bedeutet, dass die Ergebnisse eines Zeitschrittes als Input für den nächsten Zeitschritt verwendet werden. Und das bedeutet, dass jeder Fehler in der Ausgabe von Zeitschritt „J“ als Fehler in der Eingabe in Zeitschritt „K“ übernommen wird, und so weiter und so fort … was es für das Durcheinander sehr einfach macht, sich zu einer Schneeballebene zu entwickeln oder in Flammen aufzugehen. Hier sind zum Beispiel ein paar tausend Durchläufe eines Klimamodells …

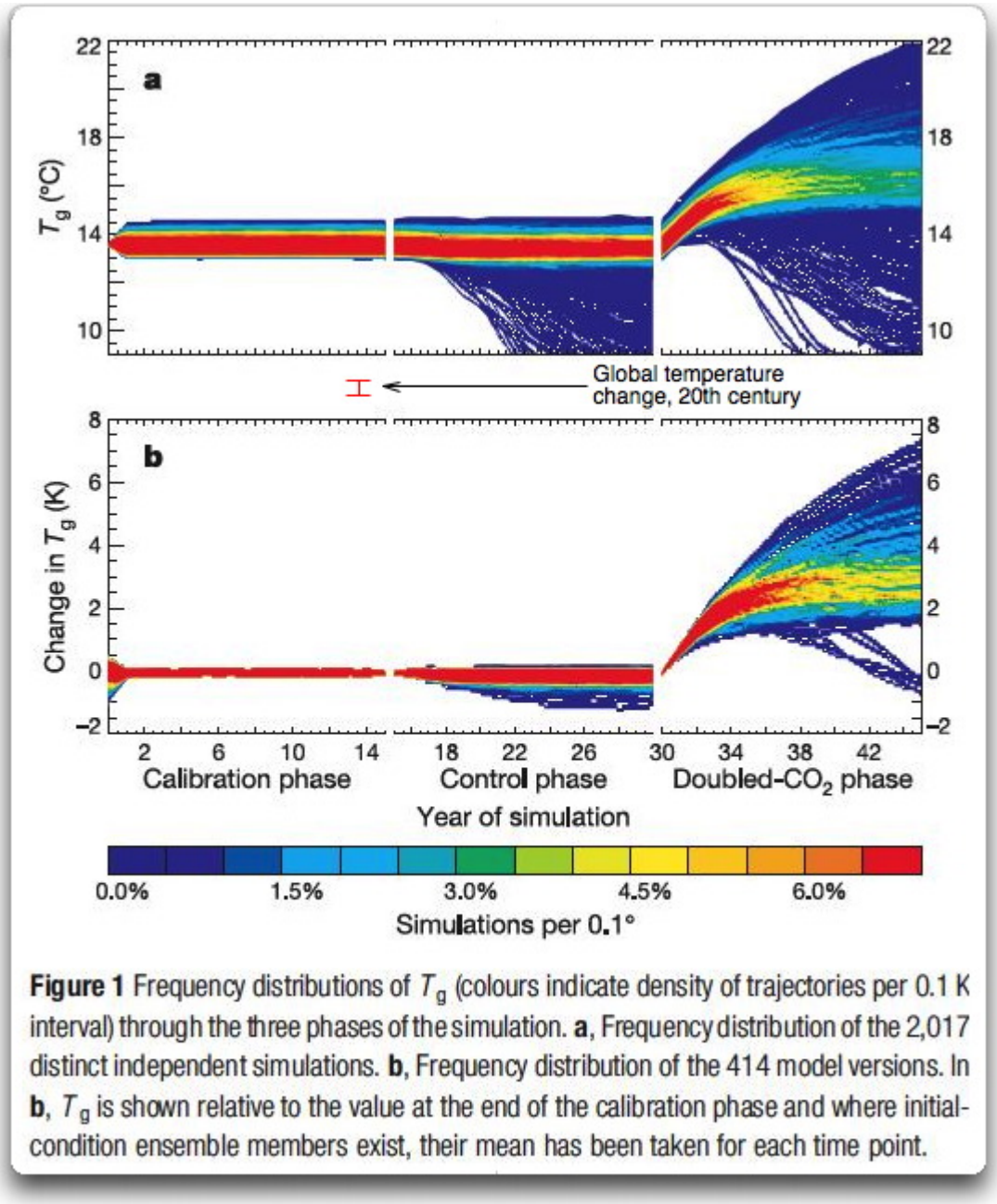

Figure 1 Frequency distributions of T_g (colours indicate density of trajectories per 0.1 K interval) through the three phases of the simulation. **a**, Frequency distribution of the 2,017 distinct independent simulations. **b**, Frequency distribution of the 414 model versions. In **b**, T_g is shown relative to the value at the end of the calibration phase and where initial-condition ensemble members exist, their mean has been taken for each time point.

Abbildung 1: 2017 Läufe des Klimamodells von www.climateprediction.net.

Man beachte im oberen Feld, wie viele Läufe während der Kontrollphase der Modellläufe nach unten fallen … und das ist bei der realen Erde noch nie passiert.

Also habe ich ein Computerprogramm geschrieben, das die 511 Einzeldateien mit den 441.688 Zeilen Computercode nach Schlüsselwörtern, Wortkombinationen und Ähnlichem durchsucht, und wenn es eine Übereinstimmung findet, listet es die Dateinummer und die Zeilennummer auf, in der das Schlüsselwort vorkommt, und gibt die betreffende Zeile und die umliegenden Zeilen aus, damit ich untersuchen kann, was es gefunden hat.

Um das zu vermeiden, hat man als Programmierer zwei Möglichkeiten. Man kann entweder beheben, was mit dem Modell nicht stimmt, das es aus den Fugen geraten lässt … oder man kann tun, was die MuddleE-Programmierer mit den Schmelztümpeln gemacht haben. Man kann einfach eine harte Grenze setzen, im Grunde eine einstellbare Leitplanke, die verhindert, dass der Muddle den Hai überspringt.

Es scheint, dass sie den Code für die Schmelztümpel verbessert haben, denn ich kann die harte Grenze für die Tage, an denen sich die Schmelztümpel bilden können, nicht mehr finden. Stattdessen haben sie den folgenden Code eingefügt:

```
C**** parameters used for Schramm sea ice albedo scheme (Hansen)
 !@var AOImin,AOImax           range for seaice albedo
 !@var ASNwet,ASNdry           wet,dry snow albedo over sea ice
 !@var AMPmin                  mininimal melt pond albedo
       REAL*8 ::
 C                        VIS   NIR1   NIR2   NIR3   NIR4   NIR5
      *     AOImin(6)=(/ .05d0, .05d0, .05d0, .050d0, .05d0, .03d0/),
      *     AOImax(6)=(/ .62d0, .42d0, .30d0, .120d0, .05d0, .03d0/),
      *     ASNwet(6)=(/ .85d0, .75d0, .50d0, .175d0, .03d0, .01d0/),              *
ASNdry(6)=(/ .90d0, .85d0, .65d0, .450d0, .10d0, .10d0/),                  *     AMPmin(6)=(/
.10d0, .05d0, .05d0, .050d0, .05d0, .03d0/)
```

Dieser Code legt harte Grenzwerte für die Albedo von Meereis und Schmelztümpeln fest und gibt konstante Werte für nassen und trockenen Schnee auf dem Meereis an. Er legt die Grenzwerte und Werte für das sichtbare Licht (VIS) sowie für fünf Banden des nahen Infrarots (NIR1-5) fest.

Das bedeutet, dass es einen Code zur Berechnung der Albedo des Meereises gibt … aber manchmal liefert dieser Code unrealistische Werte. Doch anstatt herauszufinden, warum der Code falsche Werte liefert, und diese zu korrigieren, wird der falsche Wert im Klima-Wirrwarr einfach durch die entsprechenden Höchst- oder Mindestwerte ersetzt. Wissenschaft vom Feinsten.

Hier ist ein Kommentar, der einen weiteren Schmelztümpel-Spaß beschreibt:

```
C**** safety valve to ensure that melt ponds eventually disappear (Ti<-10)
        if (Ti1 .lt.-10.) pond_melt(i,j)=0.  ! refreeze
```

Ohne diesen Teil des Codes würden einige der Schmelztümpel vielleicht nie wieder gefrieren, egal wie kalt es wurde … man muss diese Art von Physik lieben: Wasser, das nicht gefriert.

Das ist es, was die Klimamodellierer meinen, wenn sie sagen, dass ihr Modell „physikalisch basiert“ ist. Sie meinen es in demselben Sinne, in dem die Produzenten eines Hollywood-Films sagen, dass er „auf einer wahren Geschichte beruht“ …

Hier ist zum Beispiel ein toller Kommentar aus dem MuddleE-Code (das „c“ oder das „!“ in einer Zeile zeigt einen Kommentar an):

```
!@sum tcheck checks for reasonable temperatures
!@auth  Ye Cheng/G. Hartke
!@ver   1.0
c ------------------------------------------------------------------
c  This routine makes sure that the temperature remains within
c  reasonable bounds during the initialization process. (Sometimes the
c  the computed temperature iterated out in left field someplace,
c  *way* outside any reasonable range.) This routine keeps the temp
c  between the maximum and minimum of the boundary temperatures.
c ------------------------------------------------------------------
```

Mit anderen Worten: Wenn die Temperatur aus den Fugen gerät … untersuchen Sie nicht, warum das so ist, und bringen Sie es in Ordnung. Stellen Sie einfach eine angemessene Temperatur ein und machen Sie weiter.

Und was ist eine angemessene Temperatur? Es stellt sich heraus, dass sie einfach die Temperatur des vorherigen Zeitschritts einstellen und weiter machen … Physik, Sie wissen schon.

Hier ist noch eine:

```
c  ucheck makes sure that the winds remain within reasonable
c    bounds during the initialization process. (Sometimes the computed
c    wind speed iterated out in left field someplace, *way* outside
c    any reasonable range.) Tests and corrects both direction and
c    magnitude of the wind rotation with altitude. Tests the total
c    wind speed via comparison to similarity theory. Note that it
c    works from the top down so that it can assume that at level (i),
c    level (i+1) displays reasonable behavior.
```

… wenn das Klima-Kuddelmuddel aus den Fugen gerät und der Wind mit

fünfhundert Meilen pro Stunde bläst, suchen Sie nicht nach dem Grund dafür. Stützen Sie es einfach ab, stellen Sie es wieder auf die Schienen und machen Sie weiter …

Dann haben wir eine andere Klasse von Nicht-Physik. Das sind abstimmbare Parameter. Hier ist eine Beschreibung aus dem oben verlinkten Artikel von Gavin Schmidt:

Das Modell ist so eingestellt (unter Verwendung des Schwellenwerts für die relative Luftfeuchtigkeit U00 für die Entstehung von Eis- und Wasserwolken), dass es sich im globalen Strahlungsgleichgewicht befindet (d. h. die Nettostrahlung am TOA liegt innerhalb von ±0,5 W/m² von Null) und eine angemessene planetarische Albedo (zwischen 29 % und 31 %) für die Kontrolllaufsimulationen aufweist.

Mit anderen Worten, die im Klimamodell simulierte Physik wird die Modellwelt nicht im Gleichgewicht halten. Also dreht man einfach am Einstellknopf und presto! Es funktioniert alles bestens! Der U00-Regler funktionierte sogar so gut, dass man zwei weitere Regler einbaute … und eine weitere harte Grenze. Aus dem Code:

```
!@dbparam U00a tuning knob for U00 above 850 mb without moist convection
!@dbparam U00b tuning knob for U00 below 850 mb and in convective regions
!@dbparam MAXCTOP max cloud top pressure
```

Schließlich werden alle Modelle einem Prozess unterzogen, den ich „evolutionäre Abstimmung“ nenne. Das ist der Prozess, bei dem eine Änderung vorgenommen wird, und dann wird das Modell an der einzigen Sache getestet, an der wir es testen können – den historischen Aufzeichnungen. Wenn das Modell die historischen Aufzeichnungen besser wiedergeben kann, wird die Änderung beibehalten. Wenn die Änderung jedoch dazu führt, dass das Modell die Vergangenheit schlechter abbildet, wird es verworfen.

Wie es in der Werbung der Börsenmakler in den USA leider gesetzlich vorgeschrieben ist, ist die Leistung der Vergangenheit keine Garantie für den zukünftigen Erfolg. Die Tatsache, dass ein Klimamodell die Vergangenheit nachzeichnen kann, sagt absolut nichts darüber aus, ob es die Zukunft erfolgreich vorhersagen kann. Dies gilt insbesondere dann, wenn das Modell durch harte Grenzwerte und einstellbare Parameter gestützt und vor dem Umkippen bewahrt wird und dann evolutionär so eingestellt wird, dass es die Vergangenheit nachzeichnet …

Was geht sonst noch vor sich? Nun, wie bei vielen derartigen Ad-hoc-Projekten ist es dazu gekommen, dass ein einziger Variablenname für zwei verschiedene Dinge in verschiedenen Teilen des Programms steht … was ein Problem sein kann oder auch nicht, aber eine gefährliche Programmierpraxis ist, die zu unsichtbaren Fehlern führen kann. (FORTRAN

unterscheidet z. B. nicht zwischen Groß- und Kleinschreibung, also ist „ss“ die gleiche Variable wie „SS“). Hier sind einige der doppelten Variablennamen:

```
SUBR    identifies after which subroutine WATER was called
SUBR    identifies where CHECK was called from
SUBR    identifies where CHECK3 was called from
SUBR    identifies where CHECK4 was called from

ss = photodissociation coefficient, indicies
SS = SIN(lat)*SIN(dec)

ns = either 1 or 2 from reactn sub
ns = either ns or 2 from guide sub i2 newfam ifam dummy variables

nn = either nn or ks from reactn sub
nn = either nn or nnn from guide sub
nn = name of species that reacts, as defined in the MOLEC file.

ndr = either ndr or npr from guide sub
ndr = either nds or ndnr from reactn sub

Mo = lower mass bound for first size bin (kg)
Mo = total mass of condensed OA at equilibrium (ug m-3)

ks = local variable to be passed back to jplrts nnr or nn array.
ks = name of species that photolyses, as defined in the MOLEC file.

i,j = dummy loop variables
I,J = GCM grid box horizontal position
```

Schließlich ist da noch die Frage der Erhaltung von Energie und Masse. Hier ist eine Möglichkeit, wie sie gehandhabt wird …:

```
C**** This fix adjusts thermal energy to conserve total energy TE=KE+PE
       finalTotalEnergy = getTotalEnergy()
       call addEnergyAsDiffuseHeat(finalTotalEnergy - initialTotalEnergy)
```

Seltsamerweise ist das Unterprogramm „*addEnergyAsDiffuseHeat*“ zweimal in verschiedenen Teilen des Programms definiert … aber ich schweife ab. Wenn die Energie nicht erhalten bleibt, wird die Differenz einfach über den gesamten Globus hinweg addiert oder subtrahiert.

Eine Art Unterprogramm wie dieses ist notwendig, weil Computer nur bis zu einer bestimmten Anzahl von Dezimalstellen genau sind. Rundungsfehler“ sind also unvermeidlich. Und ihre Methode ist nicht unvernünftig, um mit diesem unvermeidlichen Fehler umzugehen.

Vor zwanzig Jahren fragte ich Gavin Schmidt jedoch, ob er eine Art „Murphy Gauge“ für dieses Unterprogramm hätte, um das Programm anzuhalten, wenn das Energieungleichgewicht einen bestimmten Schwellenwert überschreitet. In der realen Welt ist ein „Murphy Switchgauge“ ein Messgerät, das einen Alarm auslöst, wenn ein vom Benutzer eingestellter Wert überschritten wird. So sieht ein solches Gerät aus:

Ohne ein solches Messgerät könnte das Modell entweder eine große Menge an Energie gewinnen oder verlieren, ohne dass es jemand merkt.

Gavin sagte, er habe nichts, um das Programm bei einem zu großen Energieungleichgewicht zu stoppen. Also fragte ich ihn, wie groß das Ungleichgewicht normalerweise sei. Er sagte, er wisse es nicht.

Bei dieser Reise durch den Code 20 Jahre später suchte ich also erneut nach einem solchen „Murphy Gauge“ … aber ich konnte keines finden. Ich habe das Unterprogramm „*addEnergyAsDiffuseHeat*“ und die Umgebung durchsucht und nach allen möglichen Schlüsselwörtern wie „Energie“, „kinetisch“, „potentiell“, „thermisch“ sowie nach der FORTRAN-Anweisung

„STOP“ gesucht, die den Lauf anhält, und nach „STOP_MODEL“, einem Unterprogramm, das den Modelllauf unter bestimmten Bedingungen anhält und eine diagnostische Fehlermeldung ausgibt.

In ModelE gibt es 846 Aufrufe von „STOP_MODEL“ für alle möglichen Dinge – Seen ohne Wasser, Probleme mit Dateien, „Massendiagnosefehler“, „Druckdiagnosefehler“, Sonnenzenitwinkel nicht im Bereich [0,0 bis 1,0], Endlosschleifen, Ozeanvariablen außerhalb der Grenzen, ein STOP_MODEL, das tatsächlich ausgibt: „Bitte überprüfen Sie das eine oder andere“, und meine persönlichen Favoriten, „negative Wolkendecke“ und „negative Schneehöhe“. Ich hasse es, wenn das passiert …

Und das ist alles eine sehr gute Sache. Das sind Murphy Gauges, die das Modell stoppen sollen, wenn es aus den Fugen gerät. Sie sind ein wichtiger und notwendiger Bestandteil eines jeden solchen Modells.

Aber ich konnte keinen Murphy Gauge für die Subroutine finden, die überschüssige oder unzureichende Energie nimmt und sie gleichmäßig über den Planeten verteilt. Nun, um fair zu sein, es sind 441.668 Zeilen Code, und er ist sehr schlecht kommentiert … also könnte er da sein, aber ich konnte ihn sicher nicht aufspüren.

Also … was ist die Schlussfolgerung aus all dem?

Lassen Sie mich mit meiner Erfahrung beginnen. Ich habe mein erstes Computerprogramm vor mehr als einem halben Jahrhundert geschrieben und seitdem unzählige weitere. Auf meinem Computer befinden sich zur Zeit über 2.000 Programme, die ich in der Computersprache R geschrieben habe, mit insgesamt über 230.000 Codezeilen. Ich habe mehr Computersprachen vergessen, als ich spreche, aber ich beherrsche (oder beherrschte) C/C++, Hypertalk, Mathematica (3 Sprachen), VectorScript, Basic, Algol, VBA, Pascal, FORTRAN, COBOL, Lisp, LOGO, Datacom und R. Ich habe die gesamte Computeranalyse für die ~1.000 Beiträge, die ich für WUWT geschrieben habe, durchgeführt. Ich habe Programme für alles Mögliche geschrieben, vom Testen von Blackjack-Systemen über die Bereitstellung der CAD/CAM-Dateien für den Zuschnitt der Teile für drei 80′-Fischereistahlboote bis hin zu einem Ausschreibungssystem für den Bau kompletter Häuser, zur Erstellung der Schnittmuster für den Zuschnitt und die Montage eines 15-Meter-Fahrleitungszeltes bis hin zu … nun ja, dem Programm, das ich heute geschrieben habe, um nach Schlüsselwörtern im Code für das GISS ModelE-Klimamodell zu suchen.

Was die Programmierung betrifft, weiß ich also, wovon ich spreche.

Nun zu den Modellen. Auf meinem Planeten unterscheide ich zwischen zwei Arten von Modellen. Das sind Single-Pass-Modelle und iterative Modelle. Single-Pass-Modelle nehmen eine Reihe von Eingaben entgegen, führen einige Operationen aus und erzeugen einige Ausgaben.

Iterative Modelle hingegen nehmen eine Vielzahl von Eingaben, führen einige Operationen mit ihnen durch und erzeugen einige Ausgaben … aber

im Gegensatz zu Single-Pass-Modellen werden diese Ausgaben dann als Eingaben verwendet, mit denen das Modell Operationen durchführt, und der Prozess wird immer wieder wiederholt, um eine endgültige Antwort zu erhalten.

Bei iterativen Modellen gibt es einige sehr große Herausforderungen. Erstens sind sie, wie bereits erwähnt, im Allgemeinen sehr empfindlich und sensibel. Das liegt daran, dass jeder Fehler in der Ausgabe zu einem Fehler in der Eingabe wird. Das macht sie instabil. Und wie bereits erwähnt, gibt es zwei Möglichkeiten, das zu beheben – den Code zu korrigieren oder Leitplanken einzubauen, die verhindern, dass er aus den Fugen gerät. Der richtige Weg ist, den Code zu korrigieren … was uns zur zweiten Herausforderung führt.

Die zweite Herausforderung besteht darin, dass iterative Modelle sehr undurchsichtig sind. Wettermodelle und Klimamodelle sind iterative Modelle. Klimamodelle laufen in der Regel mit einem halbstündigen Zeitschritt. Das bedeutet, wenn ein Klimamodell beispielsweise 50 Jahre in die Zukunft voraussagt, durchläuft der Computer 48 Schritte pro Tag mal 365 Tage pro Jahr mal 50 Jahre, also 876.000 Iterationen. Und wenn er eine Antwort gibt, die keinen Sinn ergibt … wie können wir dann herausfinden, wo er aus den Fugen geraten ist?

Ich möchte betonen, dass ich nicht auf dem GISS-Modell herumhacke. Die gleichen Probleme gibt es, mehr oder weniger, bei allen großen, komplexen, iterativen Modellen. Ich weise lediglich darauf hin, dass diese Modelle NICHT „Physik-basiert“ sind – sie werden gestützt und eingezäunt, damit sie nicht abstürzen.

Abschließend möchte ich sagen, dass mich ein halbes Jahrhundert Programmieren und jahrzehntelanges Studium des Klimas einige Dinge gelehrt haben:

● Alles, was ein Computermodell tun kann, ist, die Unter- und vor allem die Missverständnisse der Programmierer sichtbar zu machen und zu verherrlichen. Punkt. Wenn Sie ein Modell in dem Glauben schreiben, dass CO2 die Temperatur steuert … raten Sie mal, was dabei herauskommt?

● Wie Alfred Korzybski bekanntlich sagte: „Die Karte ist nicht das Gebiet“. Er drückte damit auf poetische Weise aus, dass Menschen oft Modelle der Realität mit der Realität selbst verwechseln. Klimamodellierer haben dieses Problem in Hülle und Fülle, denn sie diskutieren ihre Modellergebnisse viel zu oft so, als wären sie reale Fakten.

● Das Klima ist bei weitem das komplexeste System, das wir je zu modellieren versucht haben. Es umfasst mindestens sechs Teilsysteme – Atmosphäre, Biosphäre, Hydrosphäre, Lithosphäre, Kryosphäre und Elektrosphäre. Alle diese Systeme haben interne Reaktionen, Kräfte, Resonanzen und Zyklen und stehen in Wechselwirkung mit allen anderen Systemen. Das System unterliegt variablen Kräften, die sowohl von

innerhalb als auch von außerhalb des Systems kommen. Willis‘ erste Klimaregel besagt: „Im Klima ist alles mit allem anderen verbunden … was wiederum mit allem anderen verbunden ist … außer wenn es das nicht ist.“

● Wir haben gerade erst damit begonnen, das Klima zu modellieren.

● Iterativen Modellen kann man nicht trauen. Niemals. Ja, moderne Flugzeuge werden mit iterativen Modellen entworfen … aber die Konstrukteure benutzen immer noch Windkanäle, um die Ergebnisse der Modelle zu testen. Leider haben wir nichts, was einem „Windkanal“ für das Klima entspricht.

● Die erste Regel für fehlerhaften Computercode lautet: Wenn Sie einen Fehler beseitigen, schaffen Sie wahrscheinlich zwei weitere.

● Komplexität ≠ Zuverlässigkeit. Oft liefert ein einfacheres Modell bessere Antworten als ein komplexes Modell.

Unter dem Strich: Die aktuellen Computerklimamodelle sind weit davon entfernt, als Entscheidungsgrundlage für die öffentliche Politik zu dienen. Um sich davon zu überzeugen, muss man sich nur die endlose Reihe schlechter, gescheiterter, abgestürzter und einfach falscher Vorhersagen ansehen, die die Modelle gemacht haben. Schenken Sie ihnen keine Beachtung. Sie sind nicht „physikbasiert“, außer im Sinne von Hollywood, und sie sind noch lange nicht reif für die Primetime. Ihr Hauptzweck besteht darin, den unrealistischen Ängsten der Programmierer eine falsche Legitimität zu verleihen.

Und da haben Sie es, eine komplette Tour durch das Klima-Wirrwarr.

Link:
https://wattsupwiththat.com/2022/02/18/meandering-through-a-climate-muddle/

Übersetzt von Christian Freuer für das EIKE