**R code for** **Obliquity sensitivity analysis**

# The following R script conducts an obliquity sensitivity analysis

# Adapted from Levy et al. (2019).

Levy, R. H. et al. Antarctic ice-sheet sensitivity to obliquity forcing enhanced through ocean connections. *Nat. Geosci.* **12**, 132-137 (2019). https://doi.org/10.1038/s41561-018-0284-4.

# (1) Load the R package Astrochron.

# This analysis uses version 0.9. Please use versions >= 0.9.

library(astrochron)

# (2) Loading proxy data, using STG as an example.

library(readxl)

STG <- read\_excel("C:/ZZ/STG.xlsx")

# Convert time from Ma to kyr (It is not required, depending on your time unit)

STG[1]=STG[1]\*1000

# Interpolate the STG data to a 2 kyr sampling grid.

STG\_lin=linterp(STG,dt=2)

# (3) Conduct MTM evolutive power spectral analysis for the STG data.

STGpwr=eha(STG\_lin,win=400,step=10,fmax=.06,pl=2,genplot=4,ydir=-1,pad=2000,output=2)

# (4) Integrate the STG power across the 40 kyr band.

STGint=integratePower(STGpwr,npts=401,pad=2000,flow=0.023,fhigh=0.027,ydir=-1,ln=T)

# (5) Prepare the annual mean insolation gradient (IG) between the latitudes of the core sites of STG.

library(readxl)

IG <- read\_excel("C:/ZZ//IG.xlsx")

# Resample the IG with the STG sampling grid.

IG\_resample=resample(IG,STG[,1])

# Interpolate the resampled IG to a 2 kyr sampling grid.

IG\_resample\_lin=linterp(IG\_resample,dt=2)

# (6) Conduct MTM evolutive power spectral analysis for IG.

IGPwr=eha(IG\_resample\_lin,win=400,step=10,fmax=.06,pl=2,genplot=4,ydir=-1,pad=2000,output=2)

# (7) Integrate the IG across the 40 kyr band.

IGInt=integratePower(IGPwr,npts=401,pad=2000,flow=0.023,fhigh=0.027,ydir=-1,ln=T)

# (8) Calculate obliquity sensitivity.

oblSens=cb(STGint,c(1,2))

oblSens[2]=STGint[2]/IGInt[2]

oblSens=iso(oblSens,xmin=0,xmax=4000,genplot=F)

colnames(oblSens)=c("Time\_ka","Obliquity\_Sensitivity")

# (9) Plot

autoPlot(oblSens)

**R code for** **Phase analysis**

# The following R script conducts a phase analysis

# Adapted from De Vleeschouwer et al. (2020). Phase analysis of SST-CO2 as an example.

De Vleeschouwer, D. et al. High-latitude biomes and rock weathering mediate climate-carbon cycle feedbacks on eccentricity timescales. *Nat. Commun.* **11**, 5013 (2020). https://doi.org/10.1038/s41467-020-18733-w.

setwd("C:/ZZ/Phase")

library(astrochron)

library(quantmod)

library(IRISSeismic)

rm(list=ls())

#### Here, we define the frequency ranges for the phase-analysis (in kyr-1). Not that in the manuscript, we focus on the 100-kyr eccentricity frequency range

f\_400\_low=1/0.435

f\_400\_high=1/0.370

f\_obl\_low=1/0.045

f\_obl\_high=1/0.038

f\_ecc\_low=1/0.135

f\_ecc\_high=1/0.090

# SST\_CO2 ----

SST=read.csv("SST.csv") #Reading data

CO2=read.csv("CO2.csv") #Reading data

SST\_CO2=cbind(SST,CO2[,2]) #Combining SST and CO2 data in one matrix

end=length(SST\_CO2[,1])

T1 = round(SST\_CO2[1,1], digits = 2) # T1 is the younger limit of the FIRST analysis window (sliding window approach)

if (T1 - SST\_CO2[1,1] < 0) {

if (round(T1\*100) %% 2 == 0) {T1 = T1 + 0.02}

else {T1 = T1 + 0.01}

}

if (T1 - SST\_CO2[1,1] >= 0) {

if (round(T1\*100) %% 2 == 0) {T1 = T1}

else {T1 = T1 + 0.01}

}

T2 = round(SST\_CO2[end,1], digits = 2) # T2 is the older limit of the LAST analysis window (sliding window approach)

if (T2 - SST\_CO2[end,1] < 0) {

if (round(T2\*100) %% 2 == 0) {T2 = T2}

else {T2 = T2 - 0.01}

}

if (T2 - SST\_CO2[end,1] >= 0) {

if (round(T2\*100) %% 2 == 0) {T2 = T2 - 0.02}

else {T2 = T2 - 0.01}

}

T1\_all = seq(T1, T2-0.4, by = 0.01) # T1\_all is an array that contains the younger limits of ALL analysis windows that will be applied to this site

T2\_all = seq(T1+0.4, T2, by = 0.01) # T2\_all is an array that contains the older limits of ALL analysis windows that will be applied to this site

phase\_ecc=seq(T1+0.2,T2-0.2, by = 0.01) # This array contains the midpoints (in Ma) of all analyses windows.

phase\_ecc=cbind(phase\_ecc, matrix(nrow = length(phase\_ecc), ncol = 1)) # These matrixes will be filled with phase and coherence output in the "Loop" that follows

phase\_obl=phase\_ecc

phase\_400=phase\_ecc

coh\_ecc=phase\_ecc

coh\_400=phase\_400

# Loop

for (i in 1:length(T1\_all)){

T1=T1\_all[i]

T2=T2\_all[i]

idx1=which(SST\_CO2[,1]<T2 & SST\_CO2[,1]>T1)

SST\_win1=astrochron::detrend(SST\_CO2[idx1,c(1,2)], genplot =F)

CO2\_win1=astrochron::detrend(SST\_CO2[idx1,c(1,3)], genplot = F)

res\_SST=length(SST\_win1[,1])/(T2-T1)

dt = signif(1/res\_SST,1)

time = seq(from = T1, to = T2, by = dt)

SST\_win1=approx(SST\_CO2[,1], SST\_CO2[,2], xout = time)$y

CO2\_win1=approx(SST\_CO2[,1], SST\_CO2[,3], xout = time)$y

SST\_win1=ts(SST\_win1, frequency = 1/dt, start = T1)

CO2\_win1=ts(CO2\_win1, frequency = 1/dt, start = T1)

win1=ts.union(SST\_win1,CO2\_win1) # Making time series out of the dataset in question

DF <- crossSpectrum(win1, spans=c(3,5)) # Calculating the cross spectrum

# Eccentricity ----

idx\_ecc=which(DF$freq < f\_ecc\_high & DF$freq > f\_ecc\_low) # Finding the frequency window that correspond to 100-kyr eccentricity

coh\_ecc[i,2]=max(DF$coh[idx\_ecc]) # Finding the frequency with maximum coherence within that frequency window

idx\_coh=which(DF$coh == coh\_ecc[i,2])

phase\_ecc[i,2]=DF$phase[idx\_coh] # Getting the phase result for the frequency with maximum coherence

# Obliquity ----

idx\_obl=which(DF$freq < f\_obl\_high & DF$freq > f\_obl\_low)

coh\_obl=max(DF$coh[idx\_obl])

idx\_coh=which(DF$coh == coh\_obl)

phase\_obl[i,2]=DF$phase[idx\_coh]

# # 405 kyr eccentricity ----

# idx\_400=which(DF$freq < f\_400\_high & DF$freq > f\_400\_low)

# coh\_400=max(DF$coh[idx\_400])

# idx\_coh=which(DF$coh == coh\_400)

# phase\_400[i,2]=DF$phase[idx\_coh]

}

plot(coh\_400, ylim = c(0, 1), type="l")

# Make an extra colomn in phase\_ecc that labels results with low coherency (<0.3) with "3" and high coherency (>0.6) with "1", and medium coherency with "2"

phase\_ecc=cbind(phase\_ecc, phase\_ecc[,1])

for (i in 1:length(coh\_ecc[,1])){

if (coh\_ecc[i,2]<0.3) {phase\_ecc[i,3]=3}

else if (coh\_ecc[i,2]<0.6) {phase\_ecc[i,3]=2}

else {phase\_ecc[i,3]=1}

}

# phase\_400=cbind(phase\_400, phase\_400[,1])

# for (i in 1:length(coh\_400[,1])){

# if (coh\_400[i,2]<0.3) {phase\_400[i,3]=3}

# else if (coh\_400[i,2]<0.6) {phase\_400[i,3]=2}

# else {phase\_400[i,3]=1}

# }

Site <- c("Phase")

write.csv(coh\_ecc, paste0("Phase/", Site, "\_ecc\_coh.csv"), row.names = F) # Saving results in the form of csv files

write.csv(phase\_ecc, paste0("Phase/", Site, "\_ecc.csv"), row.names = F)

#write.csv(phase\_obl, paste0("Phase/", Site, "\_obl.csv"), row.names = F)

#write.csv(phase\_400, paste0("Phase/", Site, "\_400.csv"), row.names = F)