Internship project at Alfred-Wegener-Institut

Calendar Conversion

A method to produce robust monthly and seasonal averages on orbital time scales

Carolin Krug, 13.07.2018

Contents

1 Introduction: The calendar problem

Earth's orbital parameter are changing in time [BER]. Over periods of $100,000$ and $400,000$ years, eccentricity slowly varies from (nearly) 0 to $0,0607$, inducing small changes of the annual mean total insolation received by the Earth. Obliquity oscillates from $22°$ to $25°$ over a $41,000-$year period and the position of the equinoxes precesses relative to the perihelion with $19,000-$ and $23,000-$year periods [JOUBRAC].
On an orbital timescale, these differences become relevant so that **applying today's 'fixed-day' calendar to past annual cycles can lead to significant biases.** Our aim is to define a reasonable calendar for the past which takes into account the orbital variations.
In case daily simulation data is saved, averages on a new calendar can and should be computed directly. Otherwise, we present a calendar conversion method which only needs the original (i.e. present-day-calendar) monthly means and approximates corrected monthly means.

2 Theoretical Background

2.1 Defining an angular calendar

In order to define seasons which resonate with the respective orbital configuration, it seems appropriate to compare seasonal climate with respect to earth's position along its orbit.
We consider earth's *true anomaly* ϑ: The true anomaly is defined as the angle between the perihelion on the major axis of the orbit ellipsoid and the current position of earth as shown in figure 1 and 2.

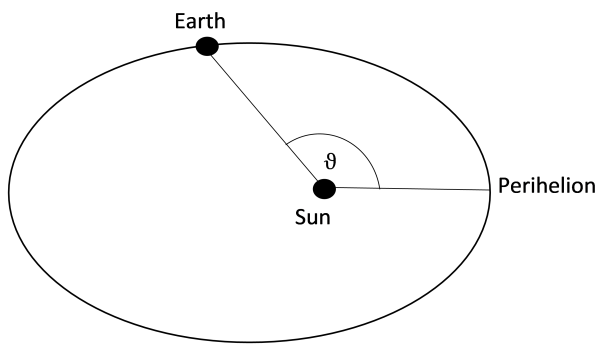

Figure 1: The true anomaly.

We then define a *month* (*season*) as a $30°$ ($90°$) increment of the true longitude, seen from a fixed starting point. Usually, the Northern Hemisphere (NH) vernal equinox (VE) is set as the beginning of spring at March 21st.

This way, we defined starting and endpoints of each month (season) via an angle. In order to obtain a calendar, we do now have to calculate month (season) lengths by calculating how much time earth needs to move from the respective starting to the endpoint of a month (season). In the following, we derive a connection between the true anomaly of any given time and the time elapsed since earth passed perihelion. We first have a look at the *mean anomaly* M, i.e. the angle between the perihelion and earth's position based on the assumption that the orbit would be a perfect circle (figure 2). The mean anomaly is defined by

$$M = \frac{2 \cdot \pi}{T} \cdot t_P \, , \tag{2.1}$$

where T is the orbital period (1 year, or 365 days) and t_P is the time elapsed since earth passed the perihelion. In order to obtain the *true anomaly* ϑ, taking into account the eccentricity ε, we define the *eccentric anomaly* E via

$$E - \varepsilon \cdot \sin(E) = M \, . \tag{2.2}$$

This equation can be solved using Newton's method. Finally, we gain the true anomaly:

$$\vartheta = 2 \cdot \arctan\left(\sqrt{\frac{1+\varepsilon}{1-\varepsilon}} \cdot \tan(\frac{e}{2})\right) \, . \tag{2.3}$$

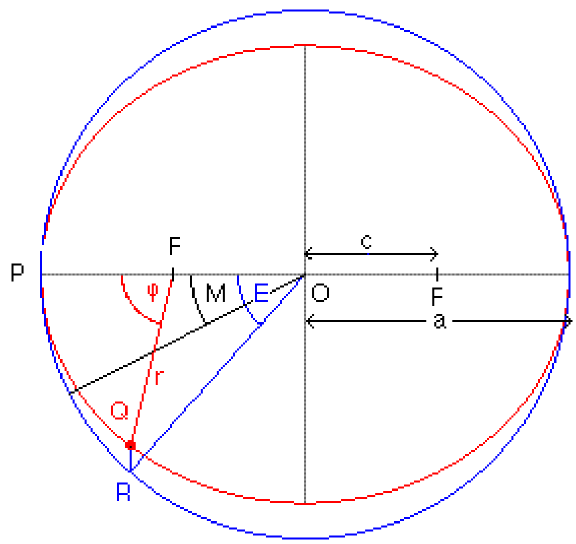

Figure 2: The mean, eccentric and true anomaly [KEP].

2.2 A calendar conversion method

If daily simulation data is available, new means can (and should) be calculated using the daily output. If not, new means have to be calculated using only the monthly means that are based on a modern calendar. For this purpose, we first have to reconstruct the daily time series in a way that original monthly (seasonal) mean averages are preserved. In the following, we present the mean preserving reconstruction algorithm.

2.2.1 The mean preserving reconstruction algorithm

The mean preserving algorithm is presented in [MP]. The basic method is nothing more than the usual running mean method with a box size (window size) of 3 days. This method leads to a smooth but not necessarily mean preserving annual cycle. In order to preserve the original means, a correction term is introduced later in section 2.2.2.

Let m be the given data set size and n the desired data set size. For our purpose of reconstructing an annual cycle out of monthly (seasonal) averages, usual $m \in \{4, 12\}$ and $n \in \{365, 366\}$. Let AVG$[m]$ be the input vector of monthly (seasonal) means and $V_{NEW}[n]$ the output vector of size m, n respectively. We initialise $V_{OLD}(i)$, $i \in \{1, \ldots, n\}$ so that each daily value equals the respective monthly mean:

$$V_{OLD}(i) = AVG(i_m)\,,$$

where $i_m \in \{1, \ldots, m\}$ is the month (season) in which day i is included.

Then, for each $i \in \{1, \ldots, n\}$, we calculate

$$V_n(i) = \frac{1}{3} \cdot \left(V_m(i-1) + V_m(i) + V_m(i+1)\right). \tag{2.4}$$

If we consider periodic data (like insolation), there are periodic boundary conditions:

$$V_{NEW}(1) = \frac{1}{3} \cdot \left(V_{OLD}(365) + V_{OLD}(1) + V_{OLD}(2)\right) \tag{2.5}$$

and

$$V_{NEW}(365) = \frac{1}{3} \cdot \left(V_{OLD}(364) + V_{OLD}(365) + V_{OLD}(1)\right). \tag{2.6}$$

If not, the endpoint values are averaged using only one neighbour:

$$V_{NEW}(1) = \frac{1}{2} \cdot \left(V_{OLD}(1) + V_{OLD}(2)\right), \tag{2.7}$$

$$V_{NEW}(365) = \frac{1}{2} \cdot \left(V_{OLD}(364) + V_{OLD}(365)\right). \tag{2.8}$$

The vector V_{OLD} is updated:

$$V_{OLD} = V_{NEW} \,. \tag{2.9}$$

After at most n iterations the algorithm converges to a stable solution [MP].

2.2.2 Mean correction term

In **every** step $i \in \{1, \ldots, n\}$ we define a mean correction term $C(m_i)$ for each month (season) $m_i \in \{1, \ldots, m\}$ by

$$C(m_i) := \mathrm{AVG}(m_i) - \frac{\sum_{i \in m_i} V_{NEW}(i)}{|m_i|} \,, \tag{2.10}$$

so that $C(m_i)$ equals the desired monthly (seasonal) average minus the current average. Adding the mean correction term to every $V_{NEW}(i)$ makes the monthly average being exactly $\mathrm{AVG}(m_i)$:
We update **all** $V_{NEW}(i), i \in \{1, \ldots, n\}$ by

$$V_{NEW}(i) = V_{NEW}(i) + C(m_i) \,. \tag{2.11}$$

In figure 3 we present the result of the mean preserving algorithm after 365 iterations.

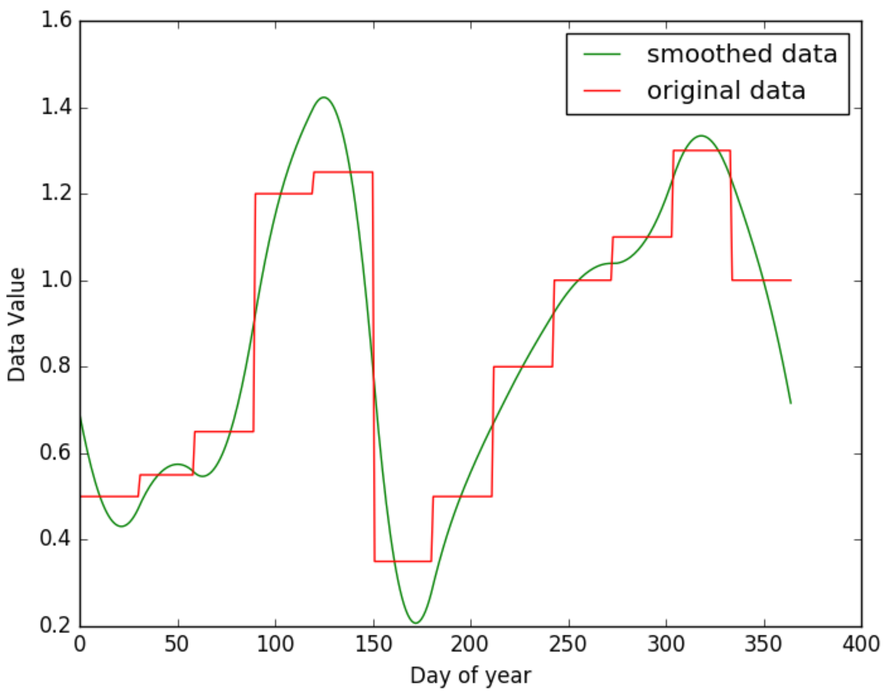

Figure 3: Reconstruction of an annual cycle of arbitrary monthly mean values (similar to [MP, p. 228]).

The mean preserving algorithm does not have to be executed for each grid point (for the usual Gaussian grid this would lead to 4608 executions per year) but only **one time** per year. For each year Y, we define a conversion matrix τ_Y which converts data from the classical calendar into Y's angular calendar. Each column i of τ_Y is obtained by applying the conversion to an annual cycle which equals 1 in month i and 0 in all other months. For details, see theorem 5.1 in the Appendix. Once we defined the conversion matrix, the conversion of any annual cycle $AVG[m]$ can be executed by multiplying by τ:

$$AVG_{NEW}[m] = \tau_Y \cdot AVG[m]\,. \tag{2.12}$$

For example, the matrix τ_{126} (rounded on two decimals) for converting monthly data from the classical calendar to a 126 ka calendar is given by:

$$\begin{pmatrix}
0.91 & -0.08 & 0.02 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01 & -0.04 & 0.19 \\
0.1 & 0.95 & -0.03 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.02 \\
0 & 0.02 & 0.99 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.01 & 1.02 & -0.04 & 0.01 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.02 & 0.08 & 1.01 & -0.09 & 0.02 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.01 & -0.04 & 0.2 & 0.94 & -0.12 & 0.03 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.02 & -0.09 & 0.45 & 0.73 & -0.13 & 0.03 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.02 & -0.09 & 0.47 & 0.71 & -0.12 & 0.02 \\
0.02 & 0 & 0 & 0 & 0 & 0 & 0 & 0.02 & -0.09 & 0.42 & 0.75 & -0.11 \\
-0.1 & 0.02 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01 & -0.06 & 0.31 & 0.83
\end{pmatrix}.$$

Considering again the arbitrary annual cycle from figure 3, the original monthly mean vector

$$v = \begin{pmatrix} 0.5 & 0.55 & 0.65 & 1.2 & 1.25 & 0.35 & 0.5 & 0.8 & 1 & 1.1 & 1.3 & 1 \end{pmatrix}$$

is converted to $v_{126} = \tau_{126} \cdot v$:

$$v_{126} = \begin{pmatrix} 0.573 & 0.545 & 0.6545 & 1.2 & 1.278 & 0.4005 & 0.436 & 0.731 & 0.948 & 1.053 & 1.263 & 1.138 \end{pmatrix}.$$

3 Implementation

3.1 General Information

The calendar conversion consists of two main parts:

1. Defining the angular calendar.
2. Computing new monthly means on the calendar.
   (a) Compute unbiased monthly means directly using daily model output.
   (b) Converting old monthly means into new monthly means (if no daily model output is available).

The function **calendar** calculates the angular calendar for given orbital parameters, the script **newmonthlymeans_daily.py** computes new monthly means using daily data while **newmonthlymeans_conv.py** converts the monthly means using the mean preserving algorithm from section 2.2.
Before describing the functions more detailed, we give an overview over some variables which appear everywhere in the code.

caltod	vector containing month (season) *lengths* for today's classical calendar
calpast	vector containing angular month (season) *lengths* for the past
timtod/timpast	vector containing *starting days* of the months (seasons) for today/past w.r.t to caltod/calpast
bias	vector containing the *offset* of calpast w.r.t caltod
VE	angle between perihelion and NH vernal equinox (in degree) (true anomaly as defined in section 2.1)
ε	earth's eccentricity

Sometimes, it is helpful to use the calendars (caltod/calpast), while sometimes you need the starting days or the calendar offset (timtod/timpast/bias). The often-used help function **calcal** returns for any given caltod and calpast the vectors timtod, timpast and bias.

3.2 The angular calendar

Function	**calendar**
Input parameter	ε, VE, desired timesteps (seasons or months)
Output	the angular calendar in integer and exact length (calpast)

For all calculations, where days are converted to angles or vice versa, we use equations (2.1), (2.2) and (2.3). Those calculations are executed using the subfunctions **day** and **ang**. For any given true longitude, day computes the time that elapsed since the perihelion, while ang computes the true longitude for any given time.
In our implementation, year length is approximated as 365 days and VE is fixed at March, 21st at noon. For calculating a monthly angular calendar, the starting day has to be shifted to the nearest start of a month, which is April 1st.
Since we cannot define April 1st as the day starting 10.5 days later than the VE (we would again apply today's calendar to a different orbital configuration, which is exactly what we want to avoid defining the angular calendar) it seems reasonable to also define April 1st via an angle. Therefore, we compute the angle between nowaday's VE and April 1st and use this as a definition for April 1st which can be used for any orbital configuration.
The months (seasons) are then defined as 30° (90°) segments of earth's orbit. The resultant lenghts are rounded using the largest remainder method: Each month (season) first gets the respective 'integer-part' number of days, the remaining days are distributed by the size of the month's (season's) decimal parts.

3.3 Processing daily simulation data

In case daily simulation data is saved, new monthly means can be computed directly.

Script	**newmonthlymeans_daily.py**
executed by	**newmean_bash_daily.sh**
Input files	12 files containing daily data for each month,
Further input parameter	variable and year
Output files	new means on the angular calendar

First, adapt the script to your needs:

1. Set the directory which contains the Input files as your working directory by redefining 'outdir'.

2. Change the input file names by redefining 'daily'.

Start the script via ./newmean_bash_daily.sh year variable (name) variable (number), for example:

./newmean_bash_daily.sh 0900 tsurf 169

The output file is safed as daily_year_variable(name).nc .

3.4 The conversion

Script	**newmonthlymeans_conv.py**
executed by	**newmean_bash_conv.sh**
Input files	12 files containing monthly data
Further input parameter	variable and year
Output files	new means using mean preserving algorithm

First, adapt the script to your needs:

1. Set the directory which contains the Input files as your working directory by redefining 'outdir'.
2. Change the input file names by redefining 'month'.

Start the script via ./newmean_bash_conv.sh year variable (name) variable (number), for example:

./newmean_bash_conv.sh 0900 tsurf 169

The output file is safed as daily_conv_variable(name).nc .

4 Results

4.1 The angular calendar

Without commenting too much we give the angular calendars and the respective orbital configuration for our test years 126 ka, 6 ka and present day (PD) as computed via function 'calendar' from section 3.2.

1. Orbital parameters

	PD	126 ka	6 ka
eccentricity	0.016724	0.039726	0.018670
VE	282.157°	112.133°	181.750°

2. Month's lengths

	PD	126 ka	6 ka
January	29	33	31
February	30	32	31
March	30	31	31
April	31	30	32
May	31	29	31
June	31	28	31
July	31	28	30
August	31	29	30
September	31	29	29
October	30	31	29
November	30	32	30
December	30	33	30

3. Season's lengths

	PD	126 ka	6 ka
Spring	93	89	93
Summer	93	85	89
Autumn	90	94	89
Winter	89	97	94

A short remark: Obviously, today's season lengths are in phase with the orbital definition of seasons, while today's calendar is **not** an angular calendar.

4.2 The conversion

In this section we want to measure the performance of our conversion method. As a measurement of how good the conversion approximates the new averages we use RMSE, or 2-norm. For $x = (x_1, \dots, x_n)$ the norm $\|x\|_2$ is defined by

$$\|x\|_2 := \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}} . \tag{4.13}$$

For each grid cell we have three vectors of monthly means. Let v_c be the vector of averages on the classical calendar, v_a the vector of averages on the angular calendar and v_m the result after our conversion method, using the mean preserving algorithm. Hence, a measurement for the exactness of the conversion method is given by

$$N_m = \|v_m - v_a\|_2 , \tag{4.14}$$

which we compare to

$$N_c = \|v_c - v_a\|_2 . \tag{4.15}$$

Above's definitions (4.14) and (4.15) are only locally, per grid cell. We gain a global estimation by averaging over all norm deviations per grid cell. In figure 4, 5 and 6, the such averaged norm deviation per year is shown for 100 years in 126 ka.

The following diagrams illustrate the method's exactness for the top incoming solar (short-wave) radiation (srad0d, figure 4), the surface temperature (tsurf, figure 5) and the surface air pressure (aps, figure 6).

We see significant differences in the performance: The conversion method performs really well for srad0d and is a clear improvement for tsurf. For a variable like aps which is more influenced by internal variability and less directly influenced by solar radiation, the method is not recommendable and we strongly suggest to keep daily data if angular calendar means are needed.

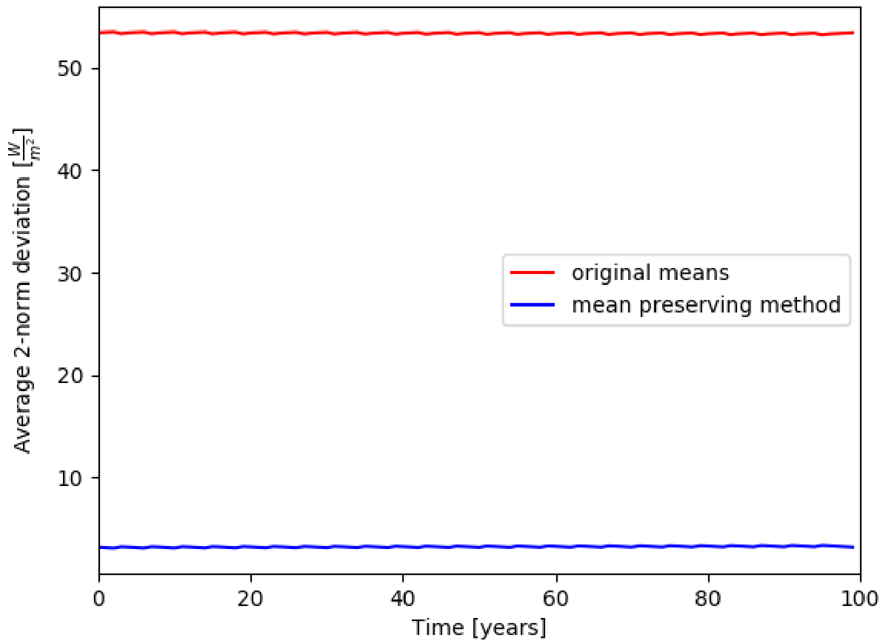

Figure 4: Comparison of original means to conversion results: srad0d

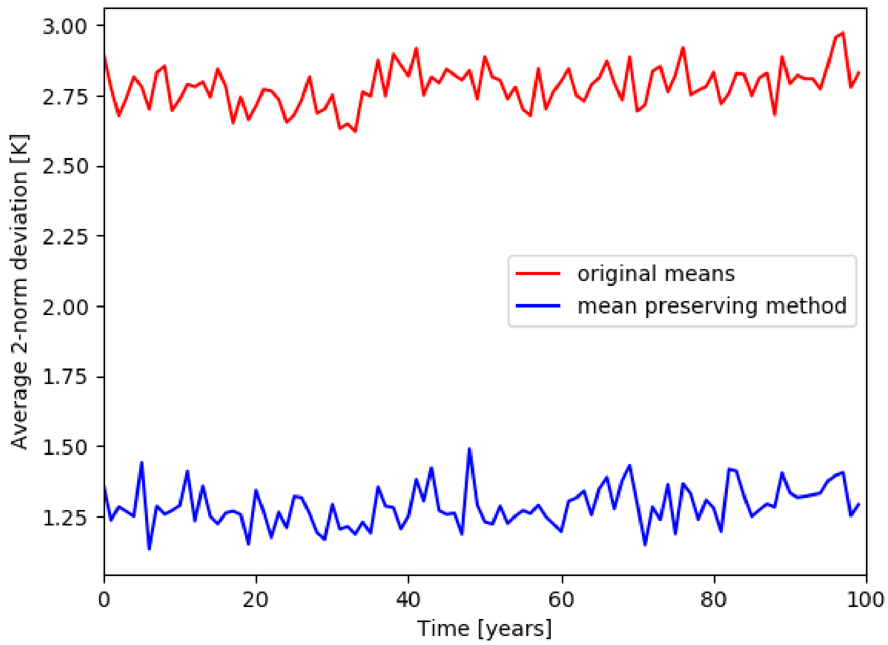

Figure 5: Comparison of original means to conversion results: tsurf

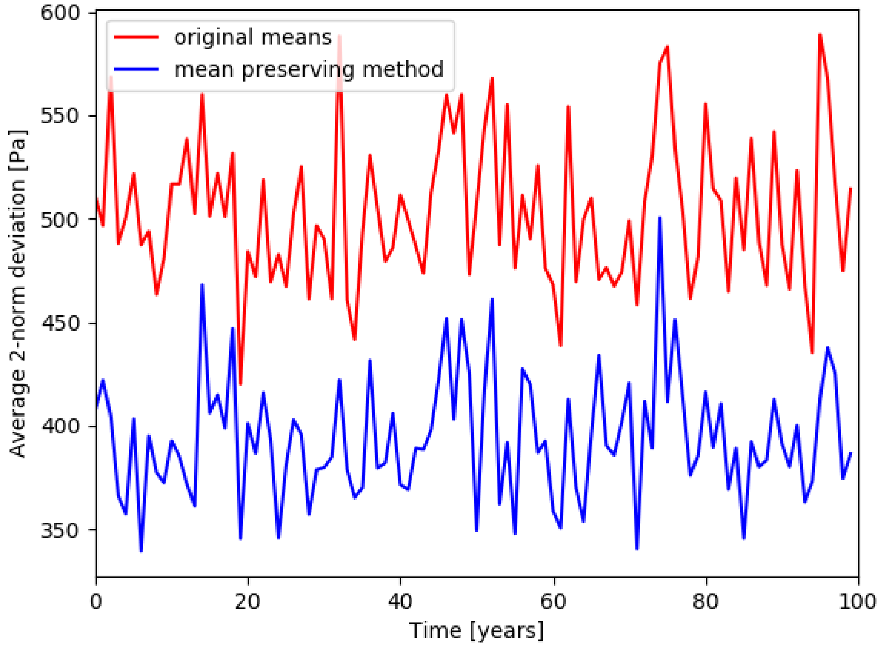

Figure 6: Comparison of original means to conversion results: aps

Finally, we have a look at the norm deviation at different regions: In figure 7 and 8 the norm deviations N_m and N_c as defined in equations (4.14) and (4.15) for an arbitrary year in 126 ka are shown. The differences between classical and angular means occur to be greater at high and mid latitudes, where the seasonal variations are bigger. After our conversion method, the results are clearly better but the effect of strong seasonal variations at high and low latitudes still exists.

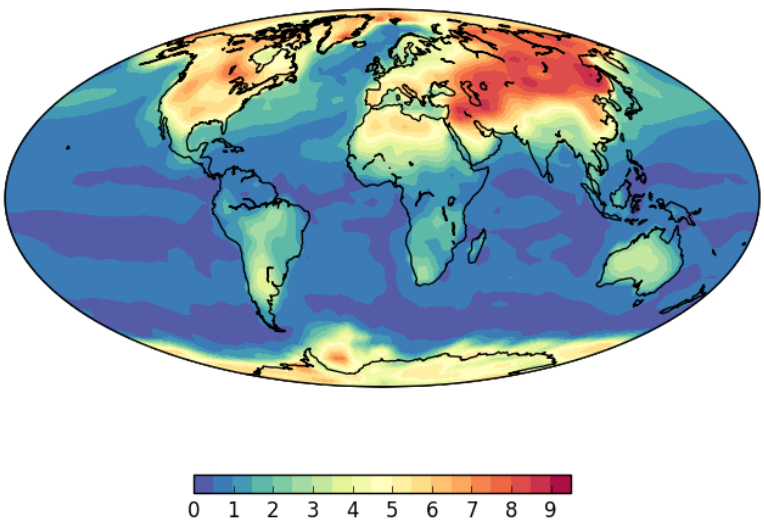

Figure 7: Comparison of original means to daily output means for variable tsurf.

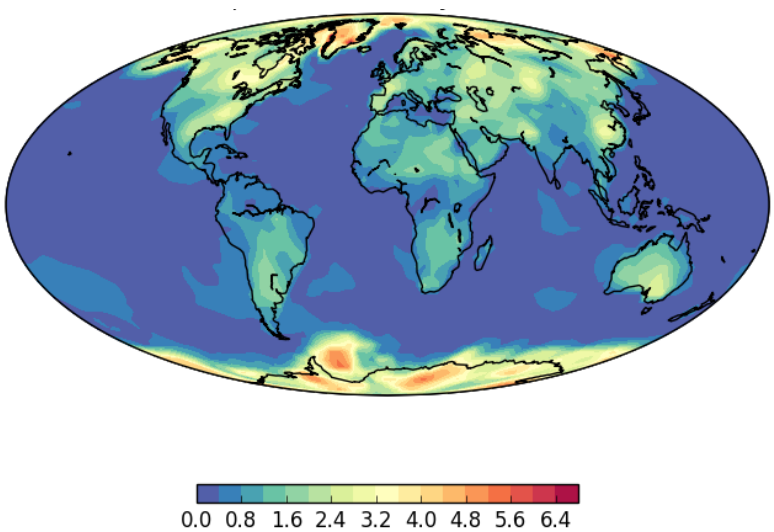

Figure 8: Comparison of conversion results to daily output means for variable tsurf.

5 Appendix

Theorem 5.1. The calendar conversion based on the mean preserving algorithm can be executed using only one climate-data-independent conversion matrix of dimension $n \times n$ per year.

Sketch of proof. Let's assume we want to convert monthly mean values, so $n = 12$. Both parts of the conversion (the reconstruction of the original cycle and the calculation of new monthly means on a new calendar) are *linear in the monthly mean values*: The annual cycle is reconstructed iteratively using equation 2.4 which is clearly linear. New means can then be calculated directly, using the reconstructed daily output, so this part is linear, too.
Then, let $\tau : \mathbb{R}^{12} \to \mathbb{R}^{12}$ be the transformation, mapping the monthly mean vector from an old calendar to the monthly mean vector on the new calendar. Since τ is linear, we know for $v \in \mathbb{R}^{12}$

$$\tau(v) = \tau(\sum_{i=1}^{12} v_i \cdot e_i) = \sum_{i=1}^{12} \tau(v_i \cdot e_i) = \sum_{i=1}^{12} \tau(e_i) \cdot v_i = \begin{pmatrix} \tau(e_1)_1 & \cdots & \tau(e_{12})_1 \\ \vdots & & \vdots \\ \tau(e_1)_{12} & \cdots & \tau(e_{12})_{12} \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_{12} \end{pmatrix}$$

So, we gain the conversion matrix $(\tau(e_j)_i)_{i,j\in\{1,\dots,12\}}$ by applying the conversion method **one time** to each unit vector $(e_1, \dots, e_{12})$.

$\square$

References

[BER] Berger, A. (1988): *Milankovitch theory and climate*, Rev. Geophys., 26, 624-657.

[EPHEM] Montenbruck, O. (2005): *Grundlagen der Ephemeridenrechnung*, Spektrum Akademischer Verlag Heidelberg, p. 55-56.

[JOUBRAC] Joussaume, S., Braconnot, P. (1997): *Sensitivity of paleoclimate simulation results to season definitions*. Journal of Geophysics Research, Vol. 102, No. D2, p. 1943-1956.

[KEP] *Solving Kepler's Equation of Elliptical Motion*, at: http://www.jgiesen.de/kepler/kepler.html (accessed at 11.07.2018).

[MP] Rymes, M. D., Myers, D. R. (2001): *Mean preserving algorithm for smoothly interpolating averaged data*, Solar Energy, Vol.71, No. 4, p. 225-231.

[POLREU] Pollard, D., Reusch. D.B. (2002): *A calendar conversion method for monthly mean paleoclimate model output with orbital forcing*. Journal of Geophysik Research, Vol. 107, No. D22, 4615, doi:10.1029/2002JD00212.