Assignment 2:
Date, time, and frequency

Questions

Dr. Robert Heinkelmann

Exercise 1

Create a function that computes 3D-rotation matrices (equations see attachment).

The first line could read

```
function [R_i]=rot3d(angle,axis)
```

where `angle` denotes the rotation angle (unit?) and `axis` specifies the rotation axis (1,2 or 3).

Both input arguments are scalar.

Output argument is a (3 x 3) matrix R_`i`.

The first line could read instead

```
function [R_i]=rot3d(axis,angle)
```

(makes no difference in terms of calculations, only different order of input variables has to be considered)

Exercise 1

function

MatLab command to define your own function

```
1 function [R_i]=rot3d(angle,axis)
```

Exercise 1

function

```
1  function [R_i]=rot3d(angle,axis)
```

List of output variables, in square brackets, comma separated. Here only 1 output variable, so no comma. In case of 1 output variable the square brackets can be omitted.

Exercise 1

function

Name of your function

```
1  function [R_i]=rot3d(angle,axis)
```

Exercise 1

function

```
1  function [R_i]=rot3d(angle,axis)
```

List of input variables in brackets, comma separated.
Brackets cannot be omitted.

Exercise 1

function

```
function [R_i]=rot3d(angle,axis)
```

Note: a function works with an own separated workspace it starts the workspace with the input variable(s) handed over by the call and initializes the variables named in brackets. When returning, the function returns only the output variable(s). Thereafter the function workspace is eliminated.

Exercise 1

function

1) Either save your function <myfun> under `<myfun>.m` in `<\mypath>`
`<myfun>` is the name of the function

here: `<myfun>.m = rot3d.m`

```
1   function [R_i]=rot3d(angle,axis)
```

This way, the function is **available from within the folder <\mypath>**.

Exercise 1

function

2) Or add your function at the end of your script <myscript>.m in <\mypath>: 'local' function.

In this case you have to end the function with an end keyword.

```
33  function [R_i]=rot3d(angle,axis)

77  end
```

This way, the function is **only defined inside this file: local function**!

Exercise 1

function

Header: create a meaningful header
that at least specifies the units
when you type `help rot3d` you get the header printed on screen

```
%% ROT3D creates a [3x3] rotation matrix
%     usage:
%     R_i = rot3d(angle,axis)
%     input:
%     angle    rotation angle [deg.dec]
%     axis     rotation axis (1, 2 or 3)
%     output:
%     R_i      [3x3] rotation marix
%     external calls:
%     none

```

Exercise 1

function Robustness: check the input variables
Here: angle, axis

```
%% robustness
if axis~=1 && axis~=2 && axis~=3
    error('2nd input argument "axis" must be 1, 2 or 3.');
end
if ~isscalar(angle)
    error('1st input argument "angle" must be scalar.');
end

```

There are probably different ways to achieve more robustness.

Exercise 1

function

1) Initialize arrays
2) Minimize computations
2) Use intrinsic functions (in principle, not for this exercise)

```
%% computations
R_i = eye(3);
sang=sind(angle);
cang=cosd(angle);
```

```
switch axis
    case 1
        R_i(2,2)=cang;
        R_i(2,3)=sang;
        R_i(3,2)=-sang;
        R_i(3,3)=cang;
    case 2
        R_i(1,1)=cang;
        R_i(1,3)=-sang;
        R_i(3,1)=sang;
        R_i(3,3)=cang;
    case 3
        R_i(1,1)=cang;
        R_i(1,2)=sang;
        R_i(2,1)=-sang;
        R_i(2,2)=cang;
end
```

Exercise 1

`script`

Use a script file to run your function
in `<\mypath>`

```
Rotation_360deg = rot3d(360,axis)
Rotation_0 = rot3d(0,axis)
Rotation_180deg_times_180deg = rot3d(180,axis)*rot3d(180,axis)
```

While executing you **specify the input variables** or directly insert **numerical values** for the input
and
you assign the **name(s) of the output variable(s)** that will be created in your workspace after the function returned.

Exercise 2

Analogue to exercise 1. No further comments.

Exercise 3

In the theory of satellite orbits, a common problem is the KEPLER equation: $M = E - e \sin E$

M: mean anomaly

E: eccentric anomaly

e: eccentricity

No problem to compute M from given E and e.

How to compute E from given M?

Exercise 3

Solution: approximation and iteration

$M = E - e \sin E$

M and e are given and are fixed.

approximate E:

$E = M + e \sin E$

iterate E or in other words: make the computation recursive: $E_{i+1} = f(E_i)$

$E_{i+1} = M + e \sin E_i$

find a suitable start value for $E_{i=0}$, here:

$E_{i=0} = M$

Note: if you have no clue for a start value, use the neutral element

Exercise 3

Solution: approximation and iteration

first evaluation:

$E_1 = M + \sin M$

is this ok? is E_1 computed precisely enough? can we stop the iteration?

→ we need a stop criterion

stop criterion here:

$|E_{i+1} - E_i| < 10^{-6}$

this stop criterion anticipates a convergent problem, i.e.

$|E_{i+2} - E_{i+1}| < |E_{i+1} - E_i| \; \forall \text{ i} \in \mathbb{N}$

Exercise 3

Evaluation: create a grid

with $M \in [0; 2\pi]$ and $e \in [0; 1[$ with resolution 0.01 for M and 0.001 for e.

```
% the grid
M_an=0:0.01:2*pi;
ecc_1=0:0.001:(1-0.001);
```

Exercise 3

compute the difference between the eccentric and mean anomalies $E - M$ and the number of iterations for each grid point

Evaluation on each grid point

```
%the quantities for each grid point
for i_M_an=1:length(M_an)
    for i_ecc_1=1:length(ecc_1)
        [E_an(i_M_an,i_ecc_1),i_ter(i_M_an,i_ecc_1)]=kepler...
            (M_an(i_M_an),ecc_1(i_ecc_1));
        E_M_diff(i_M_an,i_ecc_1)=E_an(i_M_an,i_ecc_1)-M_an(i_M_an);
    end
end
```

Exercise 4

No further comments.

Exercise 5

Write a function that provides the Julian Date and Modified Julian Date when inputting a Gregorian calendar date in year, month, day, hour, minute and second.

Exercise 5

Gregorian date to JD using the code of the Assignment#2 introductory lesson

```
function [jd,mjd] = gre2jd(yyyy,mm,dd,hour,minute,second)

% computations
fd = hour./24 + minute./1440 + second./86400;
my = fix((mm-14)./12);
jd  = fix((1461.*(yyyy + 4800 + my))./4) + ...
      fix((367.*((mm-2)-(12.*my)))./12) - ...
      fix((3.*((yyyy + 4900 +my)./100))./4) + ...
      dd - 32075.5 + fd;
mjd =  jd-2400000.5;
```

Note:
The time scale that you insert is preserved by this conversion. F.i., if you insert a date in UT1 (yyyy-mm-dd hh:mm:ss), you will get a date in UT1 (JD, MJD).

Exercise 5

1) Compute UTC from given CET

during summer time: CEST = CET + 1^h = UTC + 2^h

during winter time: CEWT=CET = UTC + 1^h

2021-11-22, $01^h{:}00^m{:}00^s$ is it summer time or winter time?

$\Rightarrow$ UTC = CET – ? = ?

2) Compute UT1 from UTC

Get dUT1 at UTC from the IERS

Exercise 5

Earth orientation data

All products are also available at the IERS FTP Server.
Plots of IERS EOP data can be configured with the IERS Plot Tool.
Click here to view the EOP of today.

Rapid data and predictions					
	Bulletin A	Plots	product metadata	latest version	available versions
Get $dUT1$ ⇨ Standard EOP data files	finals.all (IAU1980)	Plots	version metadata	latest version	
	finals.all (IAU2000)	Plots	version metadata	latest version	
	finals.data (IAU1980)	Plots	version metadata	latest version	
	finals.data (IAU2000)	Plots	version metadata	latest version	
	gpsrapid.out	Plots	version metadata	latest version	
Daily EOP data files	finals.daily (IAU1980)	Plots	version metadata	latest version	
	finals.daily (IAU2000)	Plots	version metadata	latest version	
	gpsrapid.daily	Plots	version metadata	latest version	

Monthly earth orientation data					
	Bulletin B	Plots	product metadata	latest version	available versions

Long term earth orientation data

https://www.iers.org/IERS/EN/DataProducts/EarthOrientationData/eop.html

Exercise 5

73	1	2	41684.00	I	0.120733	0.009786	0.136966	0.015902	I	$\boxed{0.8084178}$	0.0002710	0.0000	0.1916	P
73	1	3	41685.00	I	0.118980	0.011039	0.135656	0.013616	I	$\boxed{0.8056163}$	0.0002710	3.5563	0.1916	P
73	1	4	41686.00	I	0.117227	0.011039	0.134348	0.013616	I	$\boxed{0.8027895}$	0.0002710	2.6599	0.1916	P
73	1	5	41687.00	I	0.115473	0.009743	0.133044	0.013089	I	$\boxed{0.7998729}$	0.0002710	3.0344	0.1916	P
73	1	6	41688.00	I	0.113717	0.011236	0.131746	0.009898	I	$\boxed{0.7968144}$	0.0002710	3.1276	0.1916	P

this is the column for dUT1
you need only the value for one date

Exercise 5

1) Compute UTC from given CET

during summer time: CEST = CET + 1^h = UTC + 2^h

during winter time: CEWT=CET = UTC + 1^h

2021-11-22, $01^h{:}00^m{:}00^s$ is it summer time or winter time?

$\Rightarrow$ UTC = CET – ? = ?

2) Compute UT1 from UTC

Get dUT1 at UTC from the IERS

```
dUT1 = ?; % unit = s ?
```

Exercise 5

Rapid data and predictions			
Bulletin A	Plots	product metadata	latest version
finals.all (IAU1980)	Plots	version metadata	latest version

here is the info about the original unit

Year, Month, Day, Modified Julian Date

PM-x [arcsec], error_PM-x [arcsec],
PM-y [arcsec], error_PM-y [arcsec],
UT1-UTC [seconds], error_UT1-UTC [seconds],
LOD [milliseconds], error_LOD [milliseconds],
dPsi [milliarcsec], error_dPsi [milliarcsec],
dEps [milliarcsec], error_dEps [milliarcsec]

Exercise 5

2) Compute UT1 from UTC

Get dUT1 at UTC from the IERS and compute UT1 from finals all (IAU1980)

```
dUT1 = ?; % unit = s ?
```

The defining equation:

$$dUT1 = UT1 - UTC$$

$$\Rightarrow UT1 = UTC + dUT1$$

```
UT1 = UTC + dUT1 % unit = s
```

3) Convert the UT1 from (yy, mm, dd, hh, min, sec) to UT1 (JD) with your function `gre2jd`

$\mathrm{JD}_{\mathrm{UT1}}$

This input is required for both a) and b)

Exercise 5a

(a) Determine the Earth Rotation Angle (ERA) at 2021-11-22, $01^h{:}00^m{:}00^s$ CET and specify it in degree, minute and second to second precision.

Exercise 5a

$T_{UT1} = \mathrm{JD_{UT1}} - 2451545.0$

$ERA = 2 * \pi * (0.7790572732640 + 1.00273781191135448 * T_{UT1}),$

then make sure $0 \le ERA < 2\pi$

I.e. if $ERA > 2\pi$, remove complete cycles: $ERA = ERA - 2\pi$ until $ERA < 2\pi$

if $ERA < 0$, add complete cycles: $ERA = ERA + 2\pi$ until $ERA \geq 0$

Final value of ERA has to be within: $0 \le ERA < 2\pi$

Convert ERA from radiant to degree.

Then use your function from exercise#4 to convert ERA [deg] to ERA [degree,minute, second]

Finally specify ERA with second precision, i.e. `round` to integer seconds.

source: era00.for from IAU SOFA library http://www.iausofa.org/

iausofa.org

Standards of Fundamental Astronomy

Principal Sections ...

- Background
- Terms & Conditions
- Current Software
- Register
- Cookbooks
- Software Archive
- Reports & Papers
- SOFA Board

Related Links ...

- Other Implementations
- IAU Division A
- IERS
- Previous page

Acknowledge SOFA ...

If you make use of the SOFA Software, please include a citation.

Your support matters!!

The International Astronomical Union's SOFA service has the task of establishing and maintaining an accessible and authoritative set of algorithms and procedures that implement standard models used in fundamental astronomy. The service is managed by an international panel, the SOFA Board, appointed through IAU Division A — Fundamental Astronomy. SOFA also works closely with the International Earth Rotation and Reference Systems Service (IERS).

IAU SOFA Center

This web site provides access to the SOFA Software Collection which is currently available for both Fortran 77 and ANSI C. Information on how to obtain it and instructions for its use are available by following the link to Current Software.

Using SOFA software is free of charge under the terms and conditions of the SOFA licence.

Registration is encouraged as it helps to demonstrate the use being made of the SOFA Libraries and also provides users with e-mail notification of bugs and updates.

Quick start ...

Download the latest release:

- Latest Fortran 77 release is available. [Released 2021-05-12]
- Latest ANSI C release is available. [Released 2021-05-12]
- A summary of changes for the latest release is available.

Release 18 2021-05-12

GFZ Helmholtz Centre POTSDAM

TU berlin

Exercise 5b

(b) Determine the Greenwich Apparent Sidereal Time (GAST) at 2021-11-22, $01^h{:}00^m{:}00^s$ CET and specify it in hour angle, minute and second to integer second precision.

Exercise 5b

(1) Compute $GMST$ [hour]:

```
t = (JD_UT1 - 2451545.0)/36525

GMST = (F(JD_UT1)*86400 + 24110.54841 - 86400/2 + ...
        8640184.812866 * t + 0.093104 * t * t - ...
        6.2e-6 * t * t * t)/3600; % hour
```

finally $GMST$ has to be within: $0 \leq GMST < 24^{\mathrm{h}}$

The above function F reads

```
function Fx=F(x)
Fx = x - fix(x);
end
```


source: gmst82.for from IAU SOFA

Exercise 5b

Next is to compute $GAST$ [hour]:

$$GAST = GMST + Eq.E.$$

with the equation of the equinoxes [rad]

$$Eq.E. = \Delta\psi * \cos(\varepsilon_0) + (2.64 * 10^{-3} \sin\Omega + 6.3 * 10^{-5} \sin 2\Omega)\frac{\pi}{648000}$$

(2) Get $\Delta\psi$

Get nutation in longitude $\Delta\psi$ from the IERS for the given UTC date. In this case the IERS data needs to be IAU1980 compatible! (Otherwise dX, dY are listed instead of $\Delta\psi$, $\Delta\varepsilon$!)

```
delta_psi = ? ; % unit = ?
```

What is the unit of $\Delta\psi$ presented in the IERS file list?

For the above equation you need $\Delta\psi$ in the unit [rad] !

Exercise 5b

Earth orientation data

All products are also available at the IERS FTP Server.
Plots of IERS EOP data can be configured with the IERS Plot Tool.
Click here to view the EOP of today.

Rapid data and predictions

	Bulletin A	Plots	product metadata	latest version	available versions
Get $\Delta\psi$	finals.all (IAU1980)	Plots	version metadata	latest version	
Standard EOP data files	finals.all (IAU2000)	Plots	version metadata	latest version	
	finals.data (IAU1980)	Plots	version metadata	latest version	
	finals.data (IAU2000)	Plots	version metadata	latest version	
	gpsrapid.out	Plots	version metadata	latest version	
Daily EOP data files	finals.daily (IAU1980)	Plots	version metadata	latest version	
	finals.daily (IAU2000)	Plots	version metadata	latest version	
	gpsrapid.daily	Plots	version metadata	latest version	

Monthly earth orientation data

Bulletin B	Plots	product metadata	latest version	available versions

Long term earth orientation data

GFZ Helmholtz Centre POTSDAM

https://www.iers.org/IERS/EN/DataProducts/EarthOrientationData/eop.html

berlin

Exercise 5b

73	1	2	41684.00	I	0.120733	0.009786	0.136966	0.015902	I	0.8084178	0.0002710	0.0000	0.1916	P	44.969	.500
73	1	3	41685.00	I	0.118980	0.011039	0.135656	0.013616	I	0.8056163	0.0002710	3.5563	0.1916	P	45.005	.500
73	1	4	41686.00	I	0.117227	0.011039	0.134348	0.013616	I	0.8027895	0.0002710	2.6599	0.1916	P	45.122	.500
73	1	5	41687.00	I	0.115473	0.009743	0.133044	0.013089	I	0.7998729	0.0002710	3.0344	0.1916	P	45.344	.500
73	1	6	41688.00	I	0.113717	0.011236	0.131746	0.009898	I	0.7968144	0.0002710	3.1276	0.1916	P	45.623	.500

this is the column for $\Delta\psi$
you need only the value for one date

Exercise 5b

Rapid data and predictions			
Bulletin A	Plots	product metadata	latest version
finals.all (IAU1980)	Plots	version metadata	latest version

here is the info about the original unit

Year, Month, Day, Modified Julian Date

PM-x [arcsec], error_PM-x [arcsec],
PM-y [arcsec], error_PM-y [arcsec],
UT1-UTC [seconds], error_UT1-UTC [seconds],
LOD [milliseconds], error_LOD [milliseconds],
dPsi [milliarcsec], error_dPsi [milliarcsec],
dEps [milliarcsec], error_dEps [milliarcsec]

Exercise 5b

Next is to compute $GAST$ [hour]:

$$GAST = GMST + Eq.E.$$

with the equation of the equinoxes [rad]

$$Eq.E. = \Delta\psi * \cos(\varepsilon_0) + (2.64 * 10^{-3} \sin\Omega + 6.3 * 10^{-5} \sin 2\Omega) \frac{\pi}{648000}$$

(3) Compute ε_0 [rad]

$t = \frac{JD_{UT1} - 2451545.0}{36525}$ is the UT1 time since 2000-01-01, 12^h:00^m:00^s, in Julian centuries and JD_{UT1} denotes the UT1 time in Julian date format

then

$$\varepsilon_0 = (84381.448 - 46.8150\, t - 5.9 * 10^{-4}\, t^2 + 1.813 * 10^{-3}\, t^3) * \frac{\pi}{648000}$$

Exercise 5b

Next is to compute $GAST$ [hour]:

$$GAST = GMST + Eq.E.$$

with the equation of the equinoxes [rad]

$$Eq.E. = \Delta\psi * \cos(\varepsilon_0) + (2.64 * 10^{-3} \sin\Omega + 6.3 * 10^{-5} \sin 2\Omega) \frac{\pi}{648000}$$

(4) Compute Ω [rad]

$t = \frac{JD_{UT1} - 2451545.0}{36525}$ is the UT1 time since 2000-01-01, 12^h:00^m:00^s, in Julian centuries and JD_{UT1} denotes the UT1 time in Julian date format
(same as for (3))

then

$$\Omega = (450160.28 - 482890.539\, t + 7.455\, t^2 + 0.008\, t^3) \frac{\pi}{648000} + 2\pi * \mathrm{F}(-5 * t)$$

where

$\mathrm{F}(x)$ function introduced for the computation of GMST

Exercise 5b

Compute $Eq.E.$ [rad]

(5) $Eq.E. = \Delta\psi * \cos(\varepsilon_0) + (2.64 * 10^{-3} \sin \Omega + 6.3 * 10^{-5} \sin 2\Omega) \frac{\pi}{648000}$
in this equation all angles need to be specified in radiant (if you use cos and sin)

Convert $Eq.E.$ from unit [rad] into unit [hour]!

$Eq.E. = Eq.E. * \frac{12}{\pi}$

Exercise 5b

Compute $GAST$ with given $GMST$ and $Eq.E$.

(6) $GAST = GMST + Eq.E.$

GAST =
?

At the end, make sure $GAST$ is in the range $0 \leq GAST < 24$ [hour] by

$$GAST = GAST \pm n * 24$$

chose $n = \text{arbitrary integer number, so that } 0 \leq GAST < 24$

Exercise 5b

Convert $GAST$ from unit [hour] with decimal into hour angle, minute, second units and round it to second precision.

Use your function from exercise#4.

GAST = $xx^h \; yy^m \; zz^s$

Exercise 6

On a terrestrial baseline, a VLBI group delay $\Delta\tau_{TT} = 20\ \mathrm{ms}$ was observed in terrestrial time TT.

Compute the equivalent group delay in TCB, $\Delta\tau_{TCB}$, using the TT to TCG and TCG to TCB coordinate time transformations.

Exercise 6

TT to TCG coordinate time transformation for small time differences:

$$\mathrm{TCG} = \mathrm{TT} + \left(\frac{L_G}{1 - L_G}\right) \cdot (\mathrm{JD}_{\mathrm{TT}} - T_0) \cdot 86400\ \mathrm{s}$$

$$\frac{dTCG}{dTT} = 1 + \left(\frac{L_G}{1 - L_G}\right)$$

$$dTCG = \left\{1 + \left(\frac{L_G}{1 - L_G}\right)\right\} dTT$$

Exercise 6

TCG to TCB coordinate time transformation for small time differences:

$$\mathrm{TCG} = \mathrm{TCB} - \frac{1}{c^2}\left\{\int_{t_0}^{t}\left[\frac{v_e^2}{2} + V_{ext}(\vec{x}_e)\right]dt\right\} + O(c^{-4})$$

$$\frac{dTCG}{dTCB} = 1 - \frac{1}{c^2}\left[\frac{v_e^2}{2} + V_{ext}(\vec{x}_e)\right]$$

$$dTCG = \left\{1 - \frac{1}{c^2}\left[\frac{v_e^2}{2} + V_{ext}(\vec{x}_e)\right]\right\} dTCB$$

$$\Rightarrow dTCB = \frac{dTCG}{1 - \frac{1}{c^2}\left[\frac{v_e^2}{2} + V_{ext}(\vec{x}_e)\right]}$$

Exercise 6

Gravitational term

$$V_{ext}(\vec{x}_e) = \sum_{A \neq e} \frac{GM_A}{|\vec{x}_e - \vec{x}_A|}$$

Body (A)	***GM**$_A$* (m ^{3} s $^{-2}$)
Sun	$1.32712440018 \cdot 10^{20}$
Mercury	$2.2032 \cdot 10^{13}$
Venus	$3.24859 \cdot 10^{14}$
Moon	$4.9048695 \cdot 10^{12}$
Mars	$4.282837 \cdot 10^{13}$
Ceres	$6.26325 \cdot 10^{10}$
Jupiter	$1.26686534 \cdot 10^{17}$
Saturn	$3.7931187 \cdot 10^{16}$
Uranus	$5.793939 \cdot 10^{15}$
Neptune	$6.836529 \cdot 10^{15}$
Pluto	$8.71 \cdot 10^{11}$
Eris	$1.108 \cdot 10^{12}$

Exercise 6

Gravitational term

$$\sum_{A \neq e} \frac{GM_A}{|\vec{x}_e - \vec{x}_A|}$$

Body (A)	***mean distance*** (km) to Earth $|\vec{x}_e - \vec{x}_A|$
Sun	149,597,870.7 = 1 AU
Mercury	91,691,000
Venus	41,400,000
Moon	385,000.6
Mars	78,340,000
Ceres	414,000,000
Jupiter	628,730,000
Saturn	1,275,000,000
Uranus	2,723,950,000
Neptune	4,351,400,000
Pluto	5,890,000,000
Eris	96.1 AU

Exercise 6

Gravitational term sum:

$$V_{ext}(\vec{x}_e) = \sum_{A \neq e} \frac{GM_A}{|\vec{x}_e - \vec{x}_A|} = ?$$

How much do the individual gravitational terms contribute to the sum and what solar system bodies are significant for the transformation?

Individual gravitational terms:

Gravitational term of Sun:

$\frac{GM_{Sun}}{|\vec{x}_e - \vec{x}_{Sun}|}$ contributes how much to the total gravitational term (sum) in [%]

$$\left(\frac{GM_{Sun}}{|\vec{x}_e - \vec{x}_{Sun}|} \Big/ V_{ext} \right) * 100 \; [\%]$$

What is significant? → depends on the requirement of the application (here you can use 99.9%)

Exercise 6

Average magnitude of Earth barycentric velocity:

$$v_e \approx 30 \cdot 10^3 \text{ m s}^{-1}$$

Spec. relativistic term:

$$\frac{{v_e}^2}{2} = ?$$

Exercise 6

Show that the gravitational relativistic term is about twice as large as the special relativistic term.

$$V_{ext}(\vec{x}_e) \approx 2 \cdot \frac{{v_e}^2}{2} \ ?$$

Compare the numerical values on both sides of the above equation.

Exercise 6

Combine the two time transformations to get

TCB to TT time transformation in one step:

$$dTCB = \frac{dTCG}{\left\{1 - \frac{1}{c^2}\left[\frac{v_e^2}{2} + V_{ext}(\vec{x}_e)\right]\right\}} = \frac{1 + \left(\frac{L_G}{1 - L_G}\right)}{\left\{1 - \frac{1}{c^2}\left[\frac{v_e^2}{2} + V_{ext}(\vec{x}_e)\right]\right\}} dTT = ?$$

insert $\Delta\tau_{TT}$

Exercise 6

Specify the difference $(\Delta\tau_{TCB} - \Delta\tau_{TT})$ in meter unit.

What is $\Delta\tau_{TT}$=20 ms in TCB ? $\Delta\tau_{TCB}$ = ?

What is the difference $(\Delta\tau_{TCB} - \Delta\tau_{TT})$ in seconds?

Convert this time difference into meter using the speed of light c:
$\Delta baseline = (\Delta\tau_{TCB} - \Delta\tau_{TT}) \cdot c = ?$ m