Allgemeine Relativitätstheorie mit dem Computer

Aufgrund der Corona Krise findet die Vorlesung und die Übungstermine auch in diesem Semester nur Online statt.

ZOOM ONLINE MEETING
JOHANN WOLFGANG GOETHE UNIVERSITÄT
11. JUNI, 2021

MATTHIAS HANAUSKE

FRANKFURT INSTITUTE FOR ADVANCED STUDIES
JOHANN WOLFGANG GOETHE UNIVERSITÄT
INSTITUT FÜR THEORETISCHE PHYSIK
ARBEITSGRUPPE RELATIVISTISCHE ASTROPHYSIK
D-60438 FRANKFURT AM MAIN
GERMANY

9. Vorlesung

Wiederholung 8. Vorlesung

Parallele Programmierung mit OpenMP und MPI

Vorlesung 8

In der vorigen Vorlesung hatten wir in das parallele Programmieren mit C++ und OpenMP/MPI eingeführt und am Beispiel eines einfachen numerischen Problems (die Integration einer Funktion) die grundlegende Vorgehensweise eines OpenMP und MPI Programms kennengelernt. In dieser Vorlesung werden wir das numerische Lösen der Tolman-Oppenheimer-Volkoff (TOV Gleichungen, siehe Vorlesungen 6 und 7) mittels des Eulerverfahrens in einem C++ Programm durchführen und die besprochenen Parallelisierungsparadigmen (OpenMP und MPI) anwenden. Es wird sowohl ein sequentielles C++ Programm zur Berechnung der Eigenschaften von Neutronensternen, als auch eine mit OpenMP und MPI parallelisierte Version besprochen (siehe auch Teil II: Paralleles Programmieren mit C++ und OpenMP/MPI).

Sequentielles C++ Programm zur Berechnung der Eigenschaften von Neutronensternen durch numerisches Lösen der TOV Gleichungen mittels des Eulerverfahrens

Ausgehend von der, in der Vorlesung 6 hergeleiteten TOV Gleichung, wird mittels des einfachen Euler-Verfahrens die Differentialgleichung in C++ implementiert (siehe TOV sequentielle Version 2.1)). Die numerisch berechneten Werte des Neutronensternradius und seiner gravitativen Masse werden am Ende des Programms im Terminal ausgegeben. Die berechneten Resultate der Druck- und Energiedichtenprofile kann man mit den in der Vorlesung 6/7 mit Python (bzw. Maple) berechneten Ergebnissen vergleichen (siehe TOV sequentielle Version 2.1) mit Ausgabe der Ergebnisse in ein Textfile und Jupyter Notebook Die TOV-Gleichung: Zusätzliche Betrachtungen und Vergleich mit C++ Ergebnissen). Man kann zusätzlich, auf sequenzielle Weise, mehrere Neutronensterne bei festgelegter Zustandsgleichung berechnen. Dies wird einfach realisiert, indem man eine weitere for-Schleife über die zentrale Energiedichte einbaut. Der Hauptunterschied zur vorigen sequentiellen Version ist, das nun eine ganze Sequenz von Neutronensternen berechnet wird und man sich somit die Masse-Radius-Relation bei gegebener Zustandsgleichung vorstellen kann. Zusätzlich zur Version 2.1) werden auch noch die

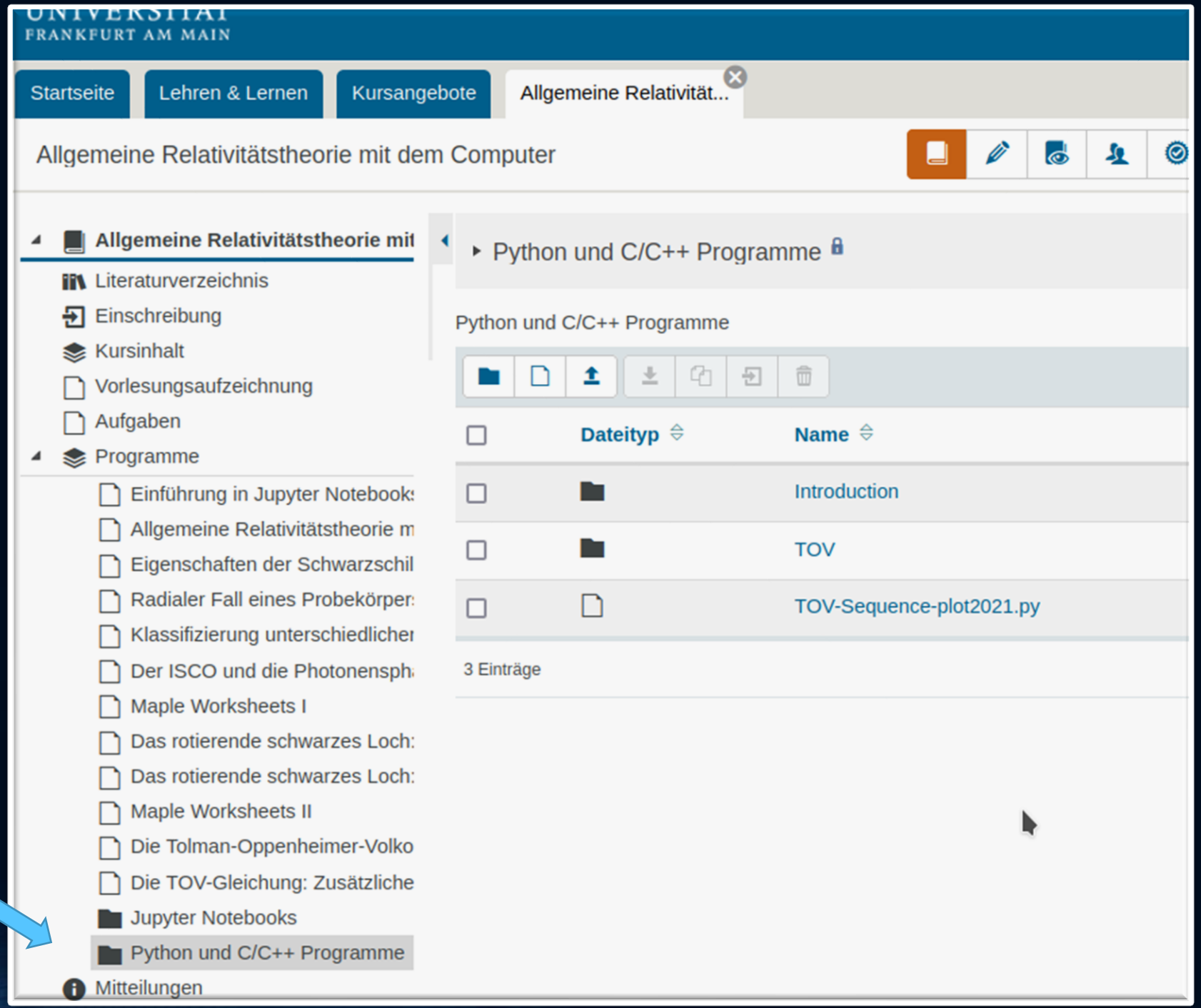
Vorlesung 8

In dieser Vorlesung werden wir uns mit dem Programmierparadigma der parallelen Programmierung befassen. Wie schon am Ende der vorigen Vorlesung erwähnt, ist es bei der Konzeption von rechenintensiven Computerprogrammen wichtig, dass die Rechenleistung des Computers stets voll ausgelastet ist und separate, voneinander unabhängige Teilaufgaben innerhalb der Programme möglichst gleichzeitig (parallel) berechnet werden. Da die meisten der großen Computerprogramme im Bereich der Allgemeinen Relativitätstheorie in der Programmiersprache C/C++ geschrieben sind, werden wir die parallele Programmierung im Folgenden am Beispiel dieser Programmiersprache verdeutlichen. Die Parallelisierung eines Python-Programms ist aber natürlich auch möglich und kann z.B. mittels des Python-Moduls (threading oder multiprocessing) implementiert werden, bzw. unter Zuhilfenahme von MPI for Python realisiert werden.

Die Programmiersprachen C/C++ und Python unterscheiden sich voneinander und bei der Erstellung der Quelldateien (source codes) eines C++ Programms ist einiges zu beachten. Jede Variable, die im Programm benutzt wir muss zunächst mit einem Typ deklariert werden (z.B. "int" für eine ganze Zahl, oder "double" für eine Fließkommazahl), mit dem Präprozessorbefehl "#include" bindet man benötigte "Header-Dateien" in das Programm ein (ähnlich den Python Bibliotheken/Modulen), die Strukturierung eines C++ Programms benötigt nicht die in Python verwendete Block-Einrückung (z.B. bei for-Schleifen), sondern verwendet geschweifte Klammern und in C/C++ muss man den Quelltext mittels eines Kompilierers in ein ausführbares Programm umwandeln. Die Art und Weise wie das parallele Programm zu konzipieren ist, hängt

Python und C/C++ Programme

Auf der OLAT Seite des Kurses finden Sie die im folgenden besprochenen C/C++ Programme



Teil II: Parallele OpenMP - Version des TOV-Programms mit geordneter Ausgabe in eine Datei und variabler Zustandsgleichung

Diese Version entspricht Version 2.5), wobei die als Funktion definierte Zustandsgleichung variabler gestaltet wurde (EOS: (e(P, K, γ)= $(P/K)^{1/\gamma}$) für Neutronensterne und Weiße Zwerge bzw. (e(P,Bag)=3 p + 4 Bag) für Quarksterne im MIT-Bag Model). Wichtig ist, dass man das Programm mit dem folgenden Befehl compiliert: 'c++ -fopenmp TOV_parallel_omp.cpp'. Führt man das Programm mit './a.out' aus, so erkennt man, dass es (in Abhängigkeit wieviele CPU-Kerne man in seinem Computer hat), viel schneller läuft.

Struktur des parallelen OpenMP-C++ Programms

```
#include <iostream>                  //Ein-/Ausgabe (Include-Dateien)
#include <math.h>                    //Mathematisches
#include <omp.h>                     //OpenMP
#include <stdio.h>                   //Fuer die Ausgabedatei

//Definition einer polytropen Zustandsgleichung
double eos(double p, double K, double gamma)
  {
    double e;
    e=pow(p/K,1.0/gamma);
```

Das parallele MPI - Programm

Teil II: Parallele MPI - Version des TOV-Programms mit geordneter Ausgabe in eine Datei und variabler Zustandsgleichung

Diese Version entspricht der OpenMP-Version Version 2.6), benutzt jedoch MPI und nicht OpenMP zur Parallelisierung und kann somit auch auf heutigen Großrechneranlagen, die über eine hohe Anzahl von Rechenknoten mit einer Vielzahl von CPU-Kernen verfügen, parallel ausgeführt werden. Wichtig ist nun, dass man das Programm mit dem folgenden Befehl compiliert: 'mpic++ TOV_parallel_omp2_eos_time.cpp' und das Programm mit 'mpirun -np 6 ./a.out' ausführt ('-np 6' ist hier nur ein Beispiel und die Zahl 6 gibt die Anzahl der Prozesse an).

Struktur des parallelen MPI-C++ Programms

```
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
#include <math.h>                      //Mathematisches
#include <stdio.h>                     //Fuer die Ausgabedatei
#include <mpi.h>                       //MPI

//Definition einer polytropen Zustandsgleichung
```

Das parallele MPI - Programm


```
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
#include <math.h>                      //Mathematisches
#include <stdio.h>                     //Fuer die Ausgabedatei
#include <mpi.h>                       //MPI

//Definition einer polytropen Zustandsgleichung
double eos(double p, double K, double gamma)
  {
    double e;
    e=pow(p/K,1.0/gamma);
    return e;
  }

//Definition einer MIT-Bag Zustandsgleichung mit cs^2=1/3 (Ueberladen der Funktion eos(...))
double eos(double p, double Bag)
  {
    double e;
    e=3.0*p + 4.0*Bag;
    return e;
  }

main( int nArguments, char **arguments )  //Hauptprogramm
  {
    MPI::Status status;
    MPI::Init(nArguments,arguments);
    int psize=MPI::COMM_WORLD.Get_size();
    int id=MPI::COMM_WORLD.Get_rank();
    printf("Prozess id= %i \n",id);
```

Das parallele MPI - Programm

Beim Ausführen des Programms spezifiziert der User mit wie vielen Prozessen er das Programm ausführen will (z.B. mit sechs Prozessen 'mpirun -np 6 ./a.out'). Ab diesem Zeitpunkt läuft das Programm mit sechs Prozessen, denen man im Laufe der weiteren Programmabfolge unterschiedliche Aufgaben zuweisen sollte, damit sie nicht alle das gleiche Ausführen. Die in der letzten Zeile angegebene Terminalausgabe erfolgt z.B. bei sechs Prozessen sechs mal; die Variable 'id' bezeichnet hier die fortlaufende Nummer des Prozesses (0,1,..,5).

```
//Variablendeklarationen
int i,anz=150;
double M,p,e,r,nu,dM,dp,de,dr,dnu,dec;
double Er[anz],EM[anz],Enu[anz],Eec[anz];
double K,gamma;
double Bag;

//Variableninitialisierung
dr=0.00001;
dec=0.00005;
gamma=5.0/3.0;
K=7.3015389;
Bag=0.0001339578066;                          //Entspricht B^(1/4)=170 MeV

//for Schleife zur Berechnung mehrerer Sterne
```

Das parallele MPI - Programm


```
//Variablendeklarationen
int i,anz=150;
double M,p,e,r,nu,dM,dp,de,dr,dnu,dec;
double Er[anz],EM[anz],Enu[anz],Eec[anz];
double K,gamma;
double Bag;

//Variableninitialisierung
dr=0.00001;
dec=0.00005;
gamma=5.0/3.0;
K=7.3015389;
Bag=0.0001339578066;                                    //Entspricht B^(1/4)=170 MeV

//for Schleife zur Berechnung mehrerer Sterne
//Aufteilung der zu berechnenden Sterne auf die einzelnen MPI-Prozesse
for (i=id;i<anz;i=i+psize)
{
```

Das parallele MPI - Programm

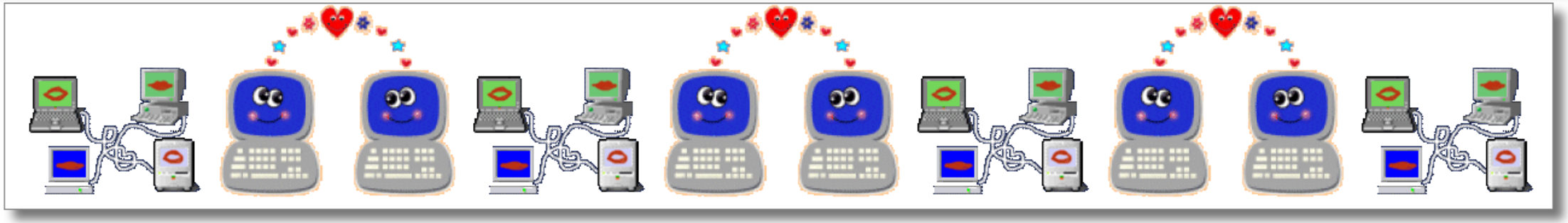
Eine sinnvolle Aufteilung der einzelnen Aufgaben des TOV-Programms auf die jeweiligen Prozesse kann man z.B. realisieren, indem die 150 (anz=150) zu berechnenden Neutronensterne auf die jeweiligen Prozesse aufteilt werden. In dem die for-Schleife von einem prozessabhängigen Startwert anfängt und in Schrittweiten 'psize' (Anzahl der Prozesse, hier z.B. 6) geht, rechnet jeder Prozess einen anderen Stern aus. Prozess 'id=0' rechnet z.B. die Sterne 'i=0,6,12,18,..' und Prozess 'id=4' rechnet z.B. die Sterne 'i=4,10,16,22,..' aus. Im Gegensatz zu den Threads in der OpenMP-Version wissen die einzelnen Prozesse der MPI Version nichts über die Berechnungen und Ergebnisse der anderen Prozesse, so dass die Werte der berechneten Ergebnisse übermittelt werden müssen - dies geschiet mit einem 'MPI::COMM_WORLD.Send(...)'-Kommando. In dieser Version senden alle Prozesse ihre Ergebnisse an Prozess mit 'id=0'.

```
    //for Schleife zur Berechnung mehrerer Sterne
    //Aufteilung der zu berechnenden Sterne auf die einzelnen MPI-Prozesse
    for (i=id;i<anz;i=i+psize)
    {
      M=0;
      r=pow(10,-14);
      p=K*pow((i+1)*dec,gamma);                    //Fuer polytrope Zustandsgleichung
//      p=1.0/3.0*(i+1)*dec-4.0/3.0*Bag;             //Fuer MIT-Bag Zustandsgleichung
      nu=0;
      Eec[i]=eos(p,K,gamma);                       //Fuer polytrope Zustandsgleichung
//      Eec[i]=eos(p,Bag);                           //Fuer MIT-Bag Zustandsgleichung

      //do-while Schleife (Numerische Lösung der TOV-Gleichung)
      do
        {
          e=eos(p,K,gamma);                        //Wert der Energiedichte bei momentanen Druck (polytrope Zustandsgleichung)
//          e=eos(p,Bag);                            //Wert der Energiedichte bei momentanen Druck (MIT-Bag Zustandsgleichung)
          dM=4*M_PI*e*r*r*dr;                      //Massenzunahme bei momentanem r und Schrittweite dr
          dp=-(p+e)*(M+4*M_PI*r*r*r*p)/(r*(r-2*M))*dr;  //Druckzunahme bei momentanem r und Schrittweite dr (TOV-Gleichung)
          dnu=(M + 4*M_PI*r*r*r*p)/(r*(r-2*M))*dr;     //Metrikzunahme bei momentanem r und Schrittweite dr
          r=r+dr;                                  //momentaner Radius des Neutronensterns
          M=M+dM;                                  //momentane Masse des Neutronensterns innerhalb des Radius r
          p=p+dp;                                  //momentaner Druck des Neutronensterns innerhalb des Radius r
          nu=nu+dnu;                               //momentane Metrik des Neutronensterns innerhalb des Radius r
        }
        while(p>0);
```

Das parallele MPI - Programm


```
    //for Schleife zur Berechnung mehrerer Sterne
    //Aufteilung der zu berechnenden Sterne auf die einzelnen MPI-Prozesse
    for (i=id;i<anz;i=i+psize)
    {
      M=0;
      r=pow(10,-14);
      p=K*pow((i+1)*dec,gamma);                          //Fuer polytrope Zustandsgleichung
//      p=1.0/3.0*(i+1)*dec-4.0/3.0*Bag;                  //Fuer MIT-Bag Zustandsgleichung
      nu=0;
      Eec[i]=eos(p,K,gamma);                             //Fuer polytrope Zustandsgleichung
//      Eec[i]=eos(p,Bag);                                //Fuer MIT-Bag Zustandsgleichung

      //do-while Schleife (Numerische Lösung der TOV-Gleichung)
      do
        {
          e=eos(p,K,gamma);                              //Wert der Energiedichte bei momentanen Druck (polytrope Zustandsgleichung)
//          e=eos(p,Bag);                                 //Wert der Energiedichte bei momentanen Druck (MIT-Bag Zustandsgleichung)
          dM=4*M_PI*e*r*r*dr;                            //Massenzunahme bei momentanem r und Schrittweite dr
          dp=-(p+e)*(M+4*M_PI*r*r*r*p)/(r*(r-2*M))*dr;   //Druckzunahme bei momentanem r und Schrittweite dr (TOV-Gleichung)
          dnu=(M + 4*M_PI*r*r*r*p)/(r*(r-2*M))*dr;       //Metrikzunahme bei momentanem r und Schrittweite dr
          r=r+dr;                                        //momentaner Radius des Neutronensterns
          M=M+dM;                                        //momentane Masse des Neutronensterns innerhalb des Radius r
          p=p+dp;                                        //momentaner Druck des Neutronensterns innerhalb des Radius r
          nu=nu+dnu;                                     //momentane Metrik des Neutronensterns innerhalb des Radius r
        }
        while(p>0);

      Er[i]=r;
      EM[i]=M;
      Enu[i]=log(1-2*M/r)/2-nu;

      //Alle Prozesse (ausser Prozess 0) senden ihre berechneten Ergebnisse an Prozess 0
      if(id != 0)
      {
        MPI::COMM_WORLD.Send( &Er[i], 1, MPI::DOUBLE, 0, 1);
        MPI::COMM_WORLD.Send( &EM[i], 1, MPI::DOUBLE, 0, 1);
        MPI::COMM_WORLD.Send( &Enu[i], 1, MPI::DOUBLE, 0, 1);
        MPI::COMM_WORLD.Send( &Eec[i], 1, MPI::DOUBLE, 0, 1);
      }

    }
```

```
  //Alle Prozesse (ausser Prozess 0) senden ihre berechneten Ergebnisse an Prozess 0
  if(id != 0)
  {
    MPI::COMM_WORLD.Send( &Er[i], 1, MPI::DOUBLE, 0, 1);
    MPI::COMM_WORLD.Send( &EM[i], 1, MPI::DOUBLE, 0, 1);
    MPI::COMM_WORLD.Send( &Enu[i], 1, MPI::DOUBLE, 0, 1);
    MPI::COMM_WORLD.Send( &Eec[i], 1, MPI::DOUBLE, 0, 1);
  }

}
```



```
//Geordnete Ausgabe der Masse, des Radius, der zentralen g00-Metrikkomponente und der zentralen Energiedichte in die Ausgabedatei
//Die Ausgabe erfolgt nur von dem Prozess 0, der zunaechst alle berechneten und an ihn gesendeten Daten empfaengt
if (id==0)
{
  for (int proc=1;proc<psize;proc++)
  {
    for (i=proc;i<anz;i=i+psize)
      {
          MPI::COMM_WORLD.Recv( &Er[i], 1, MPI::DOUBLE, proc, 1, status );
          MPI::COMM_WORLD.Recv( &EM[i], 1, MPI::DOUBLE, proc, 1, status );
          MPI::COMM_WORLD.Recv( &Enu[i], 1, MPI::DOUBLE, proc, 1, status );
          MPI::COMM_WORLD.Recv( &Eec[i], 1, MPI::DOUBLE, proc, 1, status );
       }
  }
  //Ausgabedatei
  FILE *ausgabe;
  ausgabe = fopen("output/tov.txt", "w+");          //"tov.txt" im Unterverzeichnis "output" öffnen zum speichern der Ergebnisse
```

Das parallele MPI - Programm


```
  //Geordnete Ausgabe der Masse, des Radius, der zentralen g00-Metrikkomponente und der zentralen Energiedichte in die Ausgabedatei
  //Die Ausgabe erfolgt nur von dem Prozess 0, der zunaechst alle berechneten und an ihn gesendeten Daten empfaengt
  if (id==0)
  {
    for (int proc=1;proc<psize;proc++)
    {
      for (i=proc;i<anz;i=i+psize)
        {
            MPI::COMM_WORLD.Recv( &Er[i], 1, MPI::DOUBLE, proc, 1, status );
            MPI::COMM_WORLD.Recv( &EM[i], 1, MPI::DOUBLE, proc, 1, status );
            MPI::COMM_WORLD.Recv( &Enu[i], 1, MPI::DOUBLE, proc, 1, status );
            MPI::COMM_WORLD.Recv( &Eec[i], 1, MPI::DOUBLE, proc, 1, status );
         }
    }
    //Ausgabedatei
    FILE *ausgabe;
    ausgabe = fopen("output/tov.txt", "w+");          //"tov.txt" im Unterverzeichnis "output" öffnen zum speichern der Ergebnisse

    fprintf(ausgabe,"# R[km]    M[Msol]   g00    ec[MeV/fm3] \n");
    for (i=0;i<anz;i++)
    {
      fprintf(ausgabe, "%f %f %f %e \n",Er[i],EM[i]/1.4766,exp(2*Enu[i]),Eec[i]*pow(10,6)/1.3234);
    }

    fclose(ausgabe);                                  //Ausgabedatei schliessen
  }

  MPI::Finalize ( );
  return 0;                                         //main beenden (Programmende)
}
```

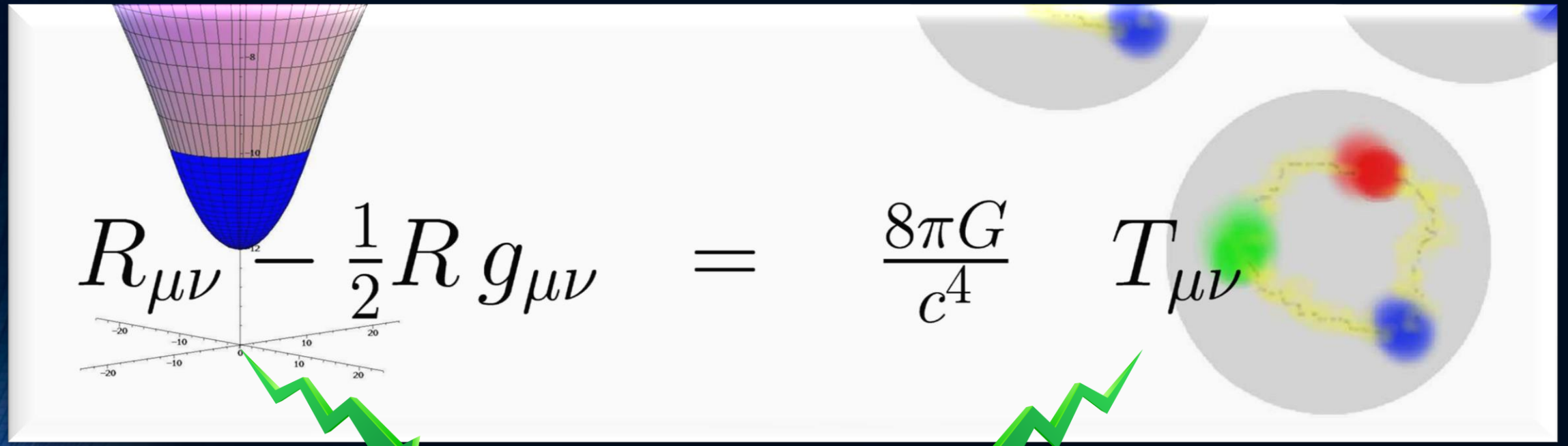
Das parallele MPI - Programm


```
    //Ausgabedatei
    FILE *ausgabe;
    ausgabe = fopen("output/tov.txt", "w+");          //"tov.txt" im Unterverzeichnis "output" öffnen zum speichern der Ergebnisse

    fprintf(ausgabe,"# R[km]    M[Msol]   g00    ec[MeV/fm3] \n");
    for (i=0;i<anz;i++)
    {
      fprintf(ausgabe, "%f %f %f %e \n",Er[i],EM[i]/1.4766,exp(2*Enu[i]),Eec[i]*pow(10,6)/1.3234);
    }

    fclose(ausgabe);                                  //Ausgabedatei schliessen
  }

  MPI::Finalize ( );
  return 0;                                         //main beenden (Programmende)
}
```

Prozess '0' empfängt dannach alle Daten und gibt diese in die Ausgabedatei aus.

Die Einstein Gleichung

Vor etwa 100 Jahren präsentierte Albert Einstein die Grundgleichung der Allgemeinen Relativitätstheorie – die sogenannte **Einstein-Gleichung**:



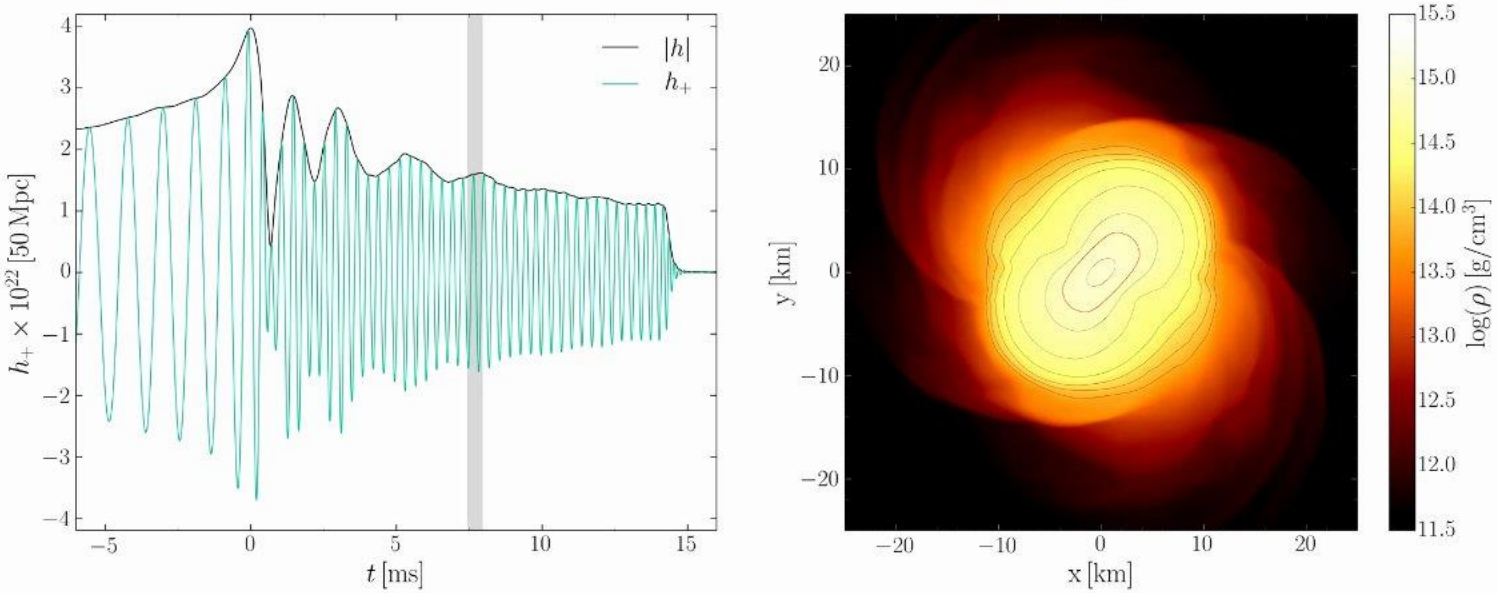
$$R_{\mu\nu} - \frac{1}{2} R\, g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

Raumzeitkrümmung
Eigenschaften der Metrik
der Raumzeit

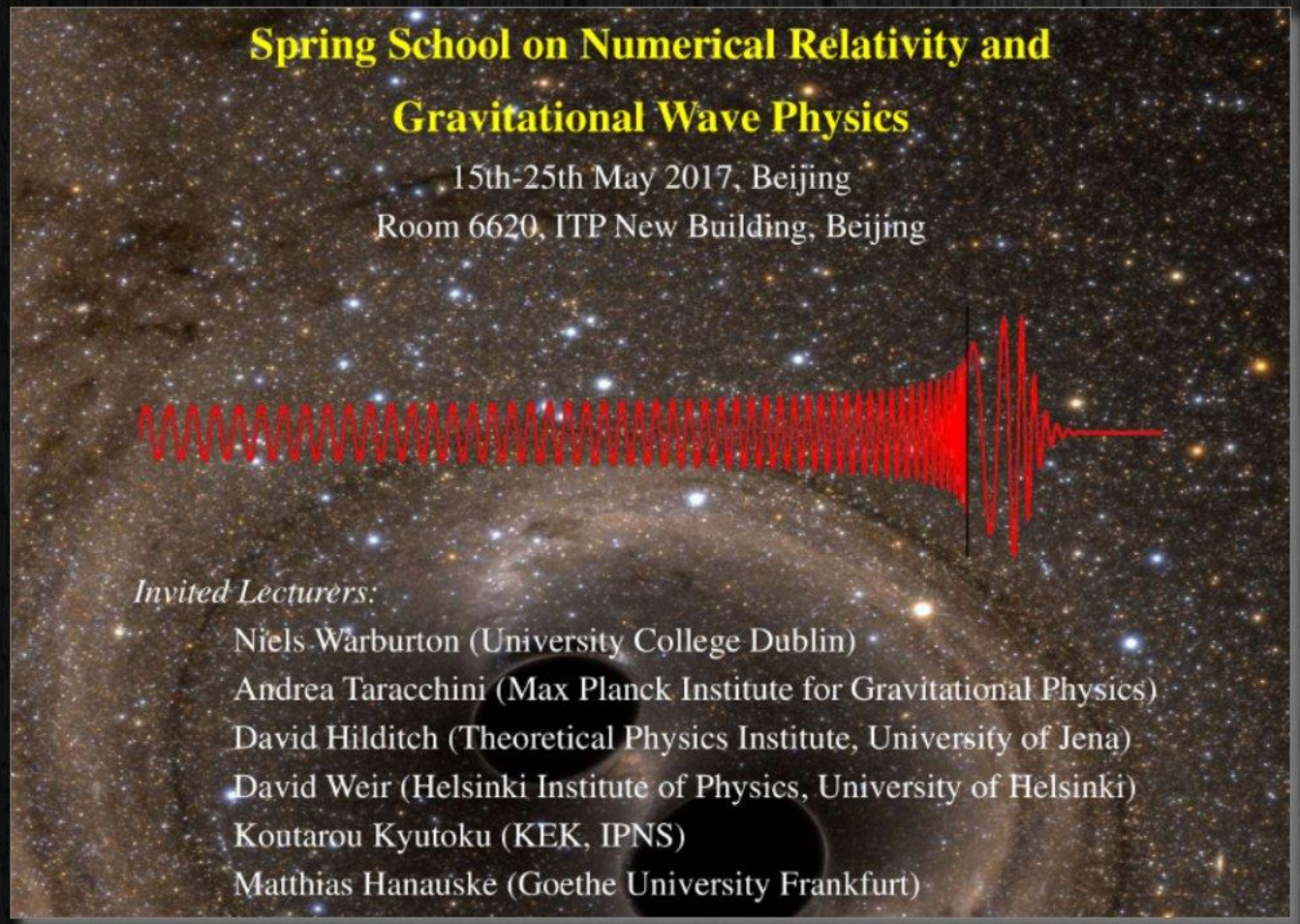
Masse, Energie und Impuls des Systems
Zustandsgleichung der Materie
Druck (Dichte , Temperatur)

Teil III: Computersimulationen mit dem Einstein-Toolkit

In diesem dritten Teil der Vorlesung soll ein Einblick in die allgemeinrelativistische Simulation auf Supercomputern gegeben werden. Unter zuhilfenahme des sogenannten Einstein-Toolkit, einer frei zugänglichen Software zur Berechnung allgemeinrelativistischer Probleme, werden unterschiedliche, realistische Systeme betrachtet. Die folgende Animation wurde mit dem Einstein-Toolkit simuliert und zeigt die emittierten Gravitationswellen und das Dichteprofil einer Neutronenstern Kollision, wobei die Animation kurz vor der Kollision startet und kurz nach dem Kollaps zum Kerr schwarzen Loch endet. Das in dieser Simulation ca. 14 Millisekunden existierende Zwischenstadium bezeichnet man als einen Hypermassiven Neutronenstern. Nachdem in den letzten Jahren mehrere Gravitationswellen zweier kollidierender schwarzer Löcher direkt nachgewiesen wurden (siehe GW150914, GW151226 und GW170104), hofft man bei der nächsten LIGO Aufzeichnung auch Gravitationswellen zweier kollidierender Neutronensterne zu finden.



Links: Amplitude der emittierte Gravitationswellen im Abstand 50 Mpc von der Kollision. Rechts: Logarithmus des Dichteprofils in der äquatorialen Ebene des Hypermassiven Neutronensterns; die rote Kontourlinie markiert den Anfang der Quarkphase im inneren Bereich des Sterns.



Die Peking Frühlingsschule 2017
Numerische Relativitätstheorie und die Physik der Gravitationswellen
(Spring School on Numerical Relativity and Gravitational Wave Physics)

Vorlesungsreihe (6 Vorlesungen) über
Gravitationswellen von kollidierenden kompakten Sternen und die Eigenschaften seltsamer Materie
(Gravitational waves from colliding compact star binaries in the context of strange/exotic matter)
致密星碰撞引起的引力波和奇异物质的性质
Beijing, China, 15.-25. May 2017

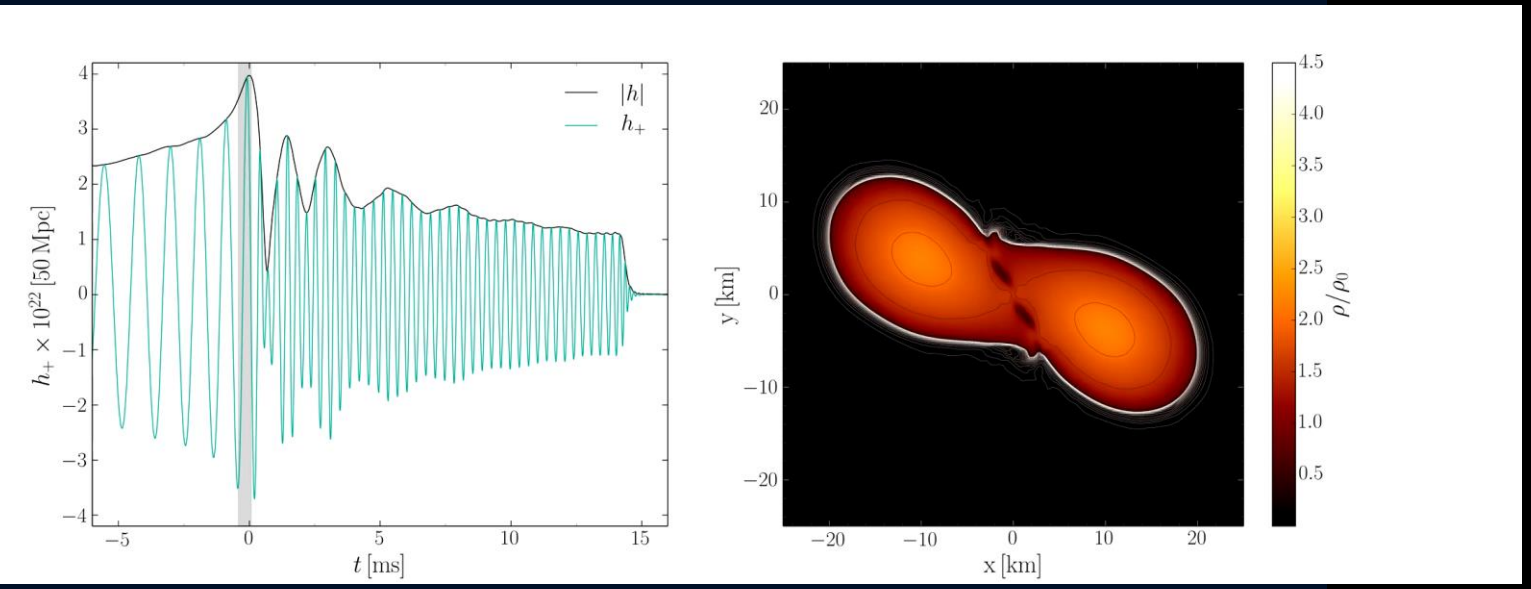
Die im Jahre 2017 gehaltene Vorlesungsreihe führt einerseits in die Allgemeine Relativitätstheorie ein, andererseits fokussiert sie sich auf den speziellen Teilaspekt der relativistischen Astrophysik kollidierender hybrider Neutronensterne, in deren innerem Bereich es zur Bildung seltsamer und exotischer Materie kommen kann. Kollabiert ein instabiler Neutronenstern zu einem schwarzen Loch oder zu einem Quark Stern? Wie kann man anhand des ausgesandten Gravitationswellen-Signals zweier kollidierender kompakter Sterne die Eigenschaften der Nuklearen- und Quark-Materie entschlüsseln?

(The series of lectures held in 2017. Topics: theory of general relativity theory, relativistic astrophysics of colliding hybrid neutron stars, strange and exotic matter in the interior of compact stars. Questions: Does an unstable neutron star collapse to a black hole or quark star? How can we extract the strange properties of high density nuclear and quark matter by means of the emitted gravitational wave signal of two colliding compact stars?)

在2017年开设的课程，一方面介绍广义相对论理论，另一方面聚焦于相对论天体物理中的一个特殊部分：混合致密星碰撞,以及在其内部可能生成的奇异和异常物质。

Teil III
Computersimulationen mit dem Einstein-Toolkit

In diesem Teil wird ein Einblick in die allgemeinrelativistische Simulation auf Supercomputern gegeben. Unter Zuhilfenahme des Einstein-Toolkits werden unterschiedliche, realistische Systeme betrachtet (z.B. Neutronenstern-Kollisionen mit Aussendung von Gravitationswellen)



Einstein Toolkit

"The Einstein Toolkit Consortium is developing and supporting open software for relativistic astrophysics. Our aim is to provide the core computational tools that can enable new science, broaden our community, facilitate interdisciplinary research and take advantage of emerging petascale computers and advanced cyberinfrastructure."

- Consortium: 94 members, 49 sites, 14 countries
- Sustainable community model:
  - 9 Maintainers from 6 sites: oversee technical developments, quality control, verification and validation, distributions and releases
  - Whole consortium engaged in directions, support, development
  - Open development meetings
  - Governance model: still being discussed (looking at CIG, iPlant)

HTTP://WWW.EINSTEINTOOLKIT.ORG

Das Einstein Toolkit: Weitere Informationen

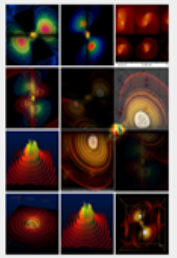
einstein toolkit

- Welcome
- About the Toolkit
  - Members
  - Maintainers
  - Governance
  - Capabilities
  - Gallery
  - Releases
  - Tools
- Download
- Community Services
  - Wiki
  - Blog
  - Support
  - Seminars
  - Issue Tracker
- Documentation
  - Tutorial for New Users
  - Citing

WELCOME

The Einstein Toolkit Consortium is developing and supporting open software for relativistic astrophysics. Our aim is to provide the core computational tools that can enable new science, broaden our community, facilitate interdisciplinary research and take advantage of emerging petascale computers and advanced cyberinfrastructure.

Please read our pages about the Einstein Toolkit, its governance, and how to get started with the toolkit for more information.



EinsteinToolkit@Flickr

Download

November 2014: We are pleased to announce the tenth release (code name "Herschel") of the Einstein Toolkit, an open, community developed software infrastructure for relativistic astrophysics.

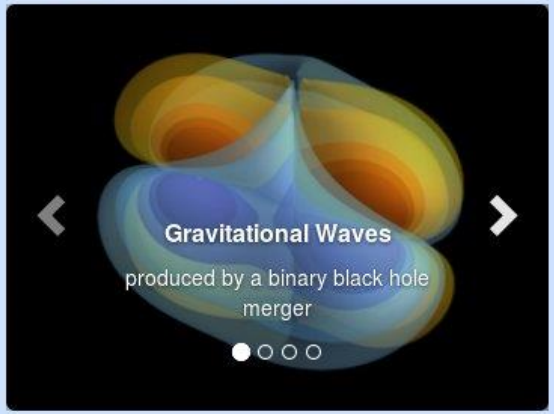
https://www.youtube.com/watch?v=EO4d32ch6OI
https://www.youtube.com/watch?v=p5bq2iUO3DE
https://www.youtube.com/watch?v=MNpyd_o0MT4
https://www.youtube.com/watch?v=Qg6PwRI2uS8
https://www.youtube.com/watch?v=ZW3aV7U-aik

Das Einstein Toolkit

einstein toolkit

Home About Download Documentation Help! Contribute Gallery

The Einstein Toolkit



Gallery

Einstein Toolkit School and Workshop

Join us at the North American Einstein Toolkit School and Workshop at NCSA, at the University of Illinois at Urbana-Champaign from July 31 to August 4 2017.

This meeting is open to anyone interested in numerical relativity and computational astrophysics and cosmology and in particular to Einstein toolkit users.

The first three days will be dedicated to a school useful for new users of the Einstein Toolkit followed by a two day long workshop open to developers interested in the Einstein Toolkit.

Registration closes *July 17, 2017*.

More information

About

The Einstein Toolkit is a community-driven software platform of core computational tools to advance and support research in relativistic astrophysics and gravitational physics.

About

Documentation

A lot of the documentation within the Einstein Toolkit is generated from comments in the source code, and more can be found on the Einstein Toolkit Wiki or other documents. We provide links to guides, tutorials and references.

Documentation

Download

We provide a convenient method to get all of the Einstein Toolkit with just a few commands, and explain the whole process.

Download

Contribute

The Einstein Toolkit would not exist without numerious contributions from its community. It is easy to learn how you can contribute as well.

Contribute

The Einstein Toolkit has been supported by NSF 1550551/1550461/1550436/1550514, NSF 1212401/1212426/1212433/1212460, NSF 0903973/0903782/0904015 (CIGR), 0701566/0855892 (XiRel), 0721915 (Alpaca), 0905046/0941653(PetaCactus/PRAC). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Das Einstein Toolkit: Download

einstein toolkit | Home | About | Download | Documentation | Help! | Contribute | Gallery

Download & Requirements

The Einstein Toolkit is hosted on many different machines around the world. We provide a script called GetComponents to simplify downloading the toolkit. This page just describes how to download the toolkit - you may also be interested in the Tutorial for New Users which leads you through these steps and more on the Queen Bee supercomputer, or in a simpler tutorial for setup on a typical Linux box.

Users of the Einstein Toolkit are encouraged to register which also signs up for the users mailing list.

Main Toolkit

Citations

The development of production level scientific software, such as the components of the Einstein Toolkit, represents the academic output of researchers. These scientific contributions should be acknowledged and respected on par with those solely based in theory or experiment. Please review our Citation Policy.

Current release: Payne-Gaposchkin (released on December 16th, 2016)

This is the recommended version of the toolkit for most users. See the release notes for more information.

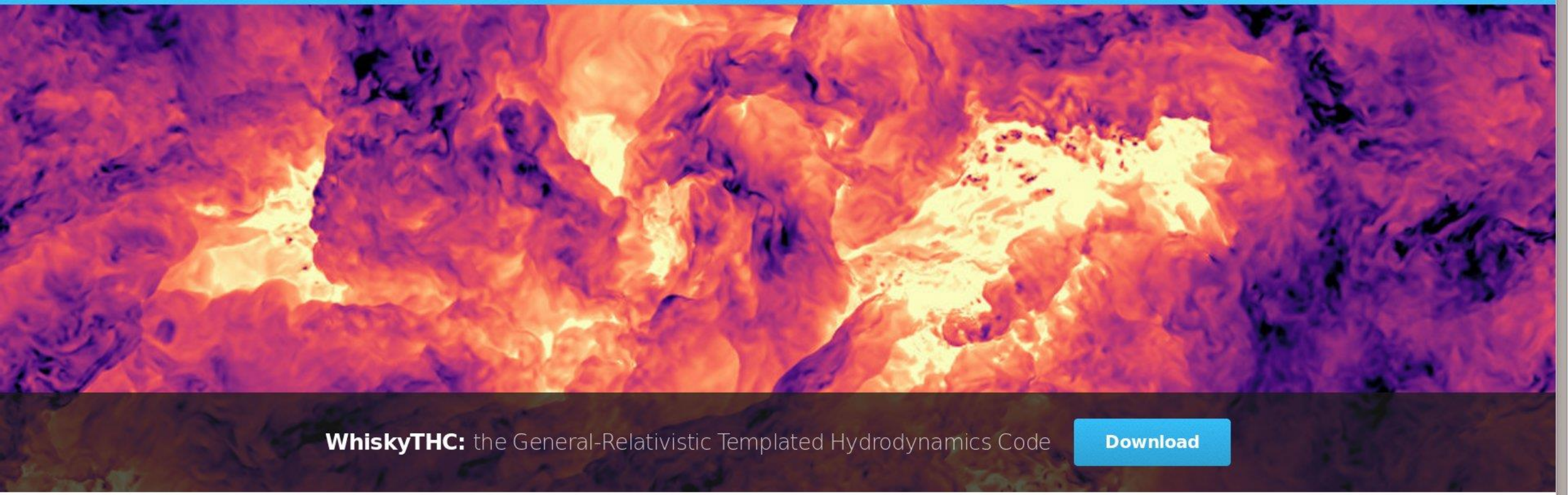
Note: OSX users cannot use the 'subversion' client shipped by Apple. In that case install subversion either from homebrew or macports.

Enter the directory on your machine in which you would like to download the ET (for example, your home directory), and type the commands listed below. This will create a directory called Cactus in which the components of the Einstein Toolkit are downloaded.

```
curl -kLO https://raw.githubusercontent.com/gridaphobe/CRL/ET_2016_11/GetComponents
chmod a+x GetComponents
./GetComponents --parallel https://bitbucket.org/einsteintoolkit/manifest/raw/ET_2016_11/einsteintoolkit.th
```

A tarball of the release is also available here, but using GetComponents is the preferred method to obtain the code. Use the tarball only if there is no way to use GetComponents (which should almost never be the case).

David Radice - Homepage



WhiskyTHC: the General-Relativistic Templated Hydrodynamics Code

Download

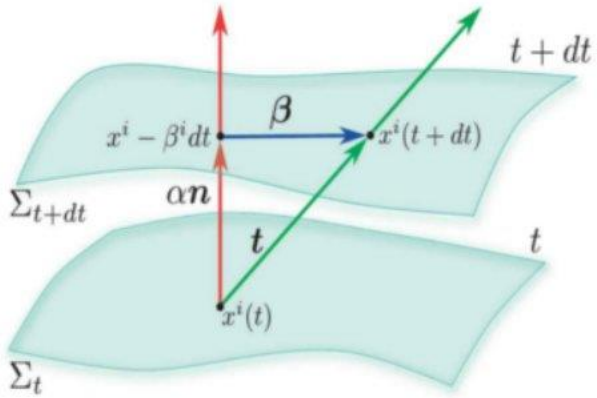
1) Die Einsteingleichung im (3+1)-Split

Die (3+1) Zerlegung der Raumzeit

$$g_{\mu\nu} = \begin{pmatrix} -\alpha^2 + \beta_i\beta^i & \beta_i \\ \beta_i & \gamma_{ij} \end{pmatrix}$$

$$x^i_{t+dt} = x^i_t - \beta^i(t, x^j)dt$$

$$d\tau^2 = \alpha^2(t, x^j)dt^2$$



Credit: L. Rezzolla, O. Zanotti: Relativistic Hydrodynamics, Oxford Univ. Press (2013)

Um die zeitliche Entwicklung von komplizierten, allgemeinrelativistischen Systemen auf dem Computer zu simulieren muss die Einsteingleichung zunächst umformuliert werden. In der sogenannten (3+1)-Zerlegung der Raumzeit wird die vierdimensionale Mannigfaltigkeit der Raumzeit in dreidimensionale, raumartige Hyperflächen Σ_t zerlegt. Die Metrik der Raumzeit $g_{\mu\nu}$ (siehe nebenstehende Abbildung) besteht in dieser Zerlegung aus einer Lapse-Funktion α, aus einem Shift-Vektor β^i und aus einer rein raumartigen Metrik γ_{ij} ($\mu\nu$=0,..3 und i,j=1..3). Die Lapse-Funktion α beschreibt den Unterschied zwischen der Koordinatenzeit t und der Eigenzeit τ und der Shift-Vektor β^i beschreibt wie stark ein Probekörper in der Raumrichtung "i" von der Struktur der Raumzeit mitgezogen wird, wenn er sich um einen infinitesimalen Zeitschritt dt bewegt. Setzt man diesen Ansatz in die Einsteingleichung ein, so gelangt man zu den sogenannten ADM-Gleichungen (nach Richard Arnowitt, Stanley Deser und Charles W. Misner), die ein System von Differentialgleichungen erster Ordnung darstellen. Um die Konvergenzeigenschaften numerischer Lösungen sicherzustellen wird im Einstein-Toolkit zusätzlich noch eine konforme, spurlose Transformation der Metrik durchgeführt und das System von Differentialgleichungen in eine hyperbolische Form gebracht. Diese Gleichungen werden dann, zusammen mit den hydrodynamischen Gleichungen, im Einstein-Toolkit numerisch gelöst.

2) Download und Kompilierung des Einstein-Toolkit

In diesem Unterpunkt werden die einzelnen Schritte beschrieben, wie man das frei erhältliche Simulationsprogramm Einstein-Toolkit installiert (für eine ausführliche Beschreibung siehe Einstein-Toolkit).

Numerical Relativity and Relativistic Hydrodynamics of Binary Neutron Star Mergers

A realistic numerical simulation of a twin star collapse, a merger of two compact stars or a collapse to a black hole needs to go beyond a static, spherically symmetric TOV-solution of the Einstein- and hydrodynamical equations.

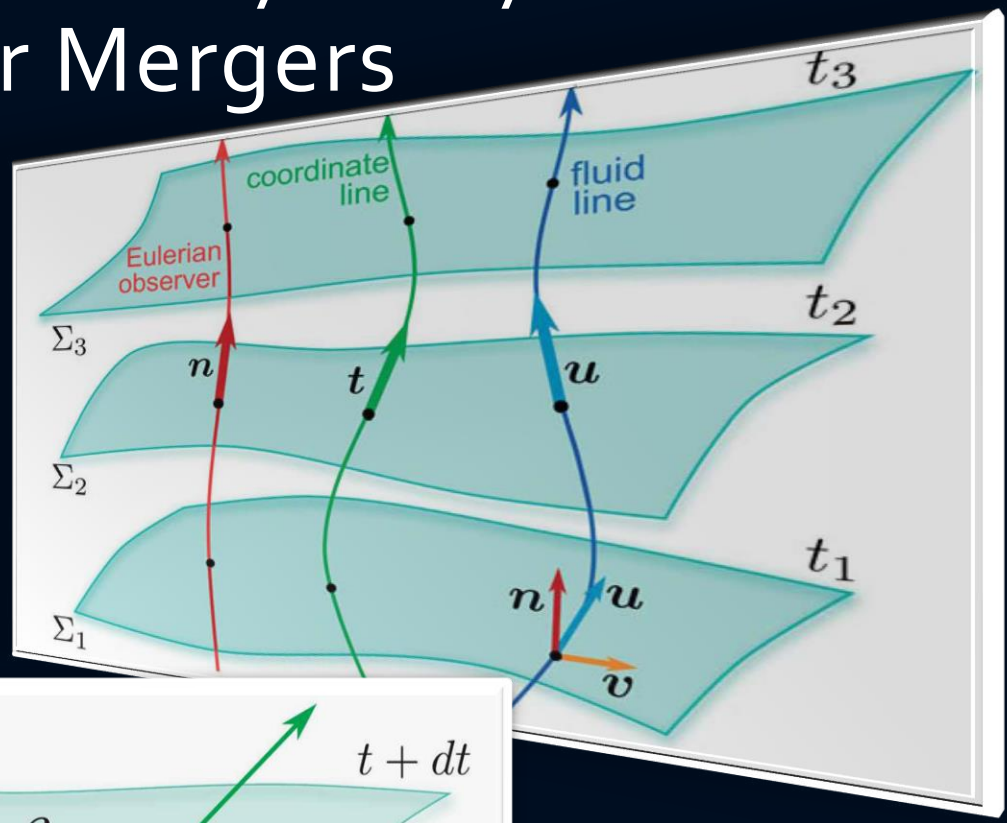
$$R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R = 8\pi T_{\mu\nu}$$

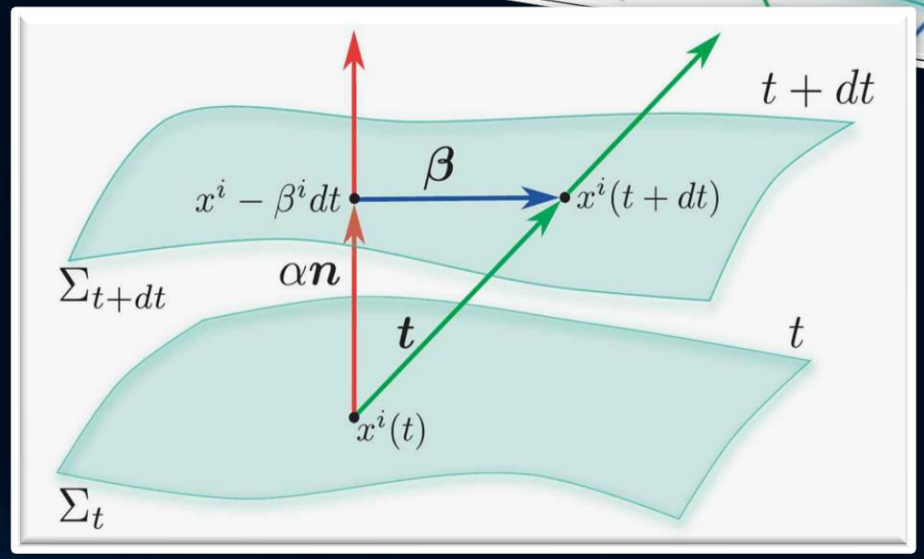
$$\nabla_\mu (\rho u^\mu) = 0\,, \qquad \nabla_\nu T^{\mu\nu} = 0\,.$$

(3+1) decomposition of spacetime

$$g_{\mu\nu} = \begin{pmatrix} -\alpha^2 + \beta_i \beta^i & \beta_i \\ \beta_i & \gamma_{ij} \end{pmatrix}$$

$$d\tau^2 = \alpha^2(t, x^j) dt^2 \qquad x^i_{t+dt} = x^i_t - \beta^i(t, x^j) dt$$





All figures and equations from: Luciano Rezzolla, Olindo Zanotti: Relativistic Hydrodynamics, Oxford Univ. Press, Oxford (2013)

The ADM equations

The ADM (Arnowitt, Deser, Misner) equations come from a reformulation of the Einstein equation using the (3+1) decomposition of spacetime.

$$\partial_t \gamma_{ij} = -2\alpha K_{ij} + \mathcal{L}_{\boldsymbol{\beta}} \gamma_{ij} = -2\alpha K_{ij} + D_i \beta_j + D_j \beta_i$$

$$\partial_t K_{ij} = -D_i D_j \alpha + \beta^k \partial_k K_{ij} + K_{ik} \partial_j \beta^k + K_{kj} \partial_i \beta^k + \alpha \left({}^{(3)}R_{ij} + K K_{ij} - 2 K_{ik} K^k{}_j \right) + 4\pi\alpha \left[\gamma_{ij} (S - E) - 2 S_{ij} \right]$$

Time evolving part of ADM

$$D_j (K^{ij} - \gamma^{ij} K) = 8\pi S^i$$

$${}^{(3)}R + K^2 - K_{ij} K^{ij} = 16\pi E$$

Constraints on each hypersurface

Three dimensional covariant derivative

$$D_\nu := \gamma^\mu{}_\nu \nabla_\mu = (\delta^\mu_\nu + n_\nu n^\mu) \nabla_\mu$$

Three dimensional Riemann tensor

$${}^{(3)}R^\mu{}_{\nu\kappa\sigma} = \partial_\kappa {}^{(3)}\Gamma^\mu_{\nu\sigma} - \partial_\sigma {}^{(3)}\Gamma^\mu_{\nu\kappa} + {}^{(3)}\Gamma^\mu_{\lambda\kappa} {}^{(3)}\Gamma^\lambda_{\nu\sigma} - {}^{(3)}\Gamma^\mu_{\lambda\sigma} {}^{(3)}\Gamma^\lambda_{\nu\kappa}$$

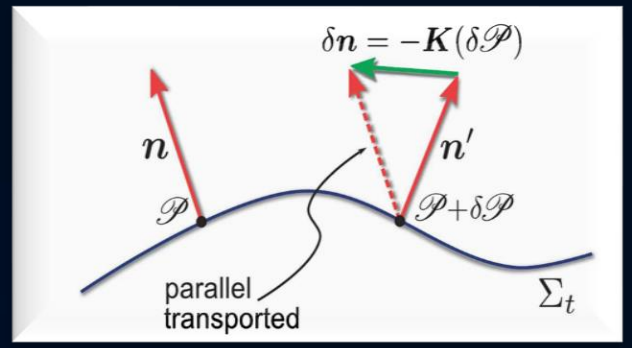
$${}^{(3)}\Gamma^\alpha_{\beta\gamma} = \frac{1}{2} \gamma^{\alpha\delta} \left(\partial_\beta \gamma_{\gamma\delta} + \partial_\gamma \gamma_{\delta\beta} - \partial_\delta \gamma_{\beta\gamma} \right)$$

Spatial and normal projections of the energy-momentum tensor:

$$S_{\mu\nu} := \gamma^\alpha{}_\mu \gamma^\beta{}_\nu T_{\alpha\beta}\,,$$
$$S_\mu := -\gamma^\alpha{}_\mu n^\beta T_{\alpha\beta}\,,$$
$$S := S^\mu{}_\mu\,,$$
$$E := n^\alpha n^\beta T_{\alpha\beta}\,,$$

Extrinsic Curvature:

$$K_{\mu\nu} := -\gamma^\lambda{}_\mu \nabla_\lambda n_\nu$$



All figures and equations from: Luciano Rezzolla, Olindo Zanotti: Relativistic Hydrodynamics, Oxford Univ. Press, Oxford (2013)

From ADM to BSSNOK

Unfortunately the ADM equations are only weakly hyperbolic (mixed derivatives in the three dimensional Ricci tensor) and therefore not "well posed". It can be shown that by using a conformal traceless transformation, the ADM equations can be written in a hyperbolic form. This reformulation of the ADM equations is known as the BSSNOK (Baumgarte, Shapiro, Shibata, Nakamuro, Oohara, Kojima) formulation of the Einstein equation. Most of the numerical codes use this (or even better the CCZ4) formulation.

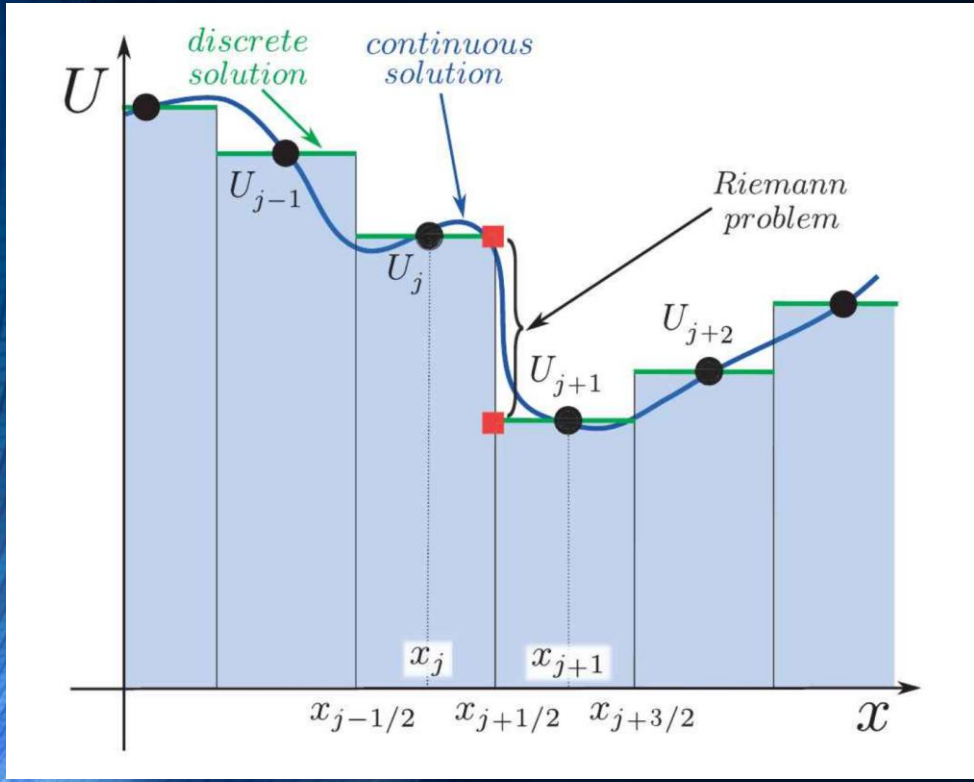
The 3+1 Valencia Formulation of the Relativistic Hydrodynamic Equations

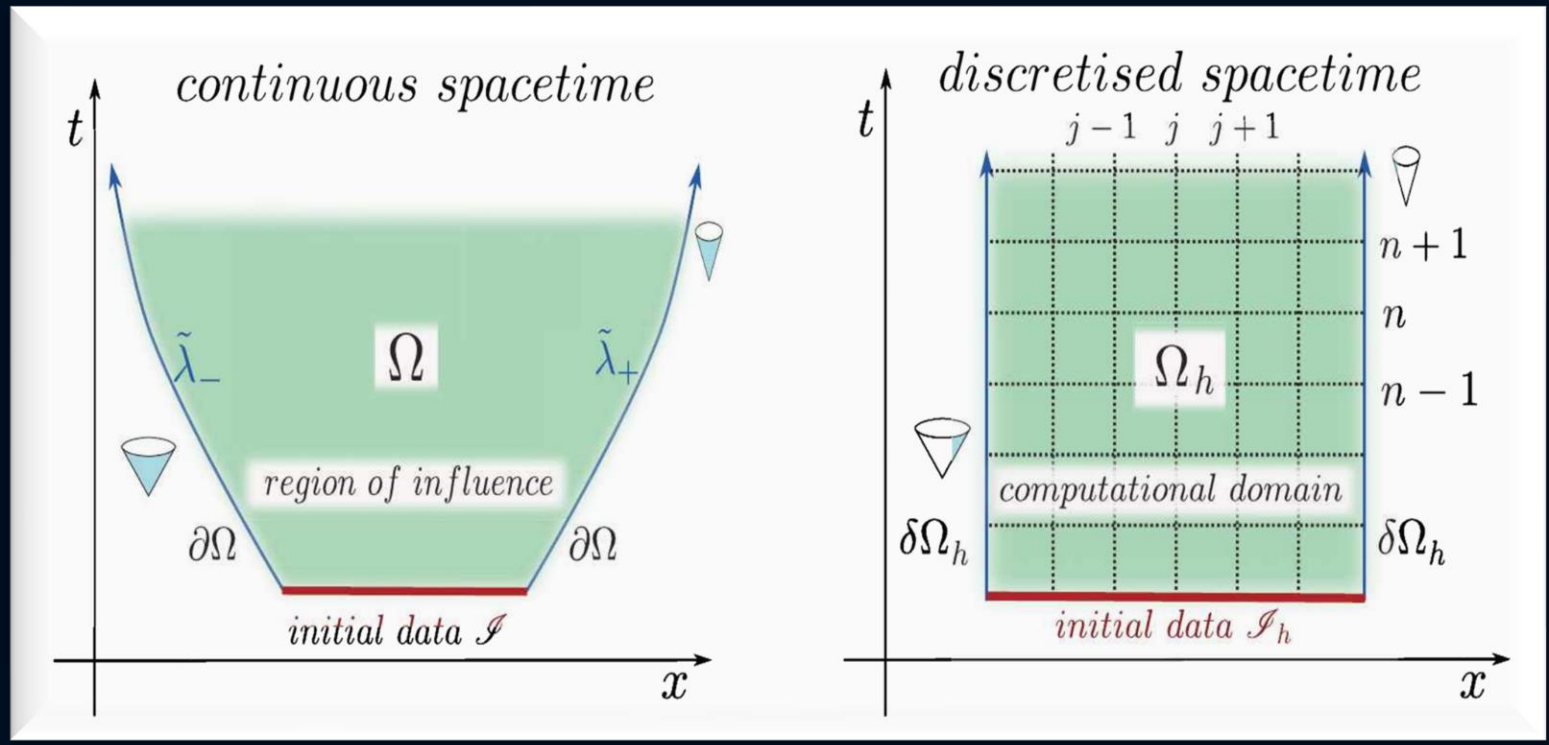
$$\nabla_\mu(\rho u^\mu) = 0\,,$$
$$\nabla_\nu T^{\mu\nu} = 0\,.$$

To guarantee that the numerical solution of the hydrodynamical equations (the conservation of rest mass and energy-momentum) converge to the right solution, they need to be reformulated into a conservative formulation. Most of the numerical “hydro codes” use here the 3+1 Valencia formulation.

Finite difference methods

Discretisation of a hyperbolic initial value boundary problem.





High resolution shock capturing methods (HRSC methods) are needed, when Riemann problems of discontinuous properties and shocks needs to be evolved accurately.

All figures from: Luciano Rezzolla, Olindo Zanotti: Relativistic Hydrodynamics, Oxford Univ. Press, Oxford (2013)

Gauge Conditions

On each spatial hypersurface, four additional degrees of freedom need to be specified: A slicing condition for the lapse function and a spatial shift condition for the shift vector need to be formulated to close the system. In an optimal gauge condition, singularities should be avoided and numerical calculations should be less time consuming.

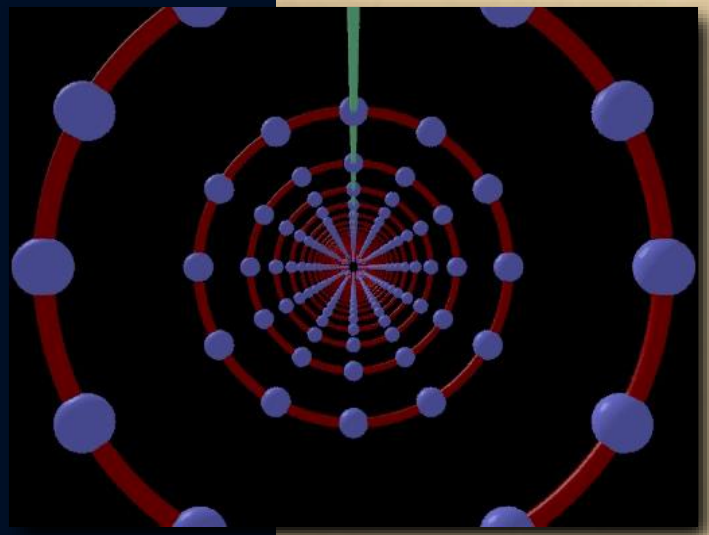
Bona-Massó family of slicing conditions: $$\partial_t \alpha - \beta^k \partial_k \alpha = -f(\alpha)\,\alpha^2 (K - K_0)$$

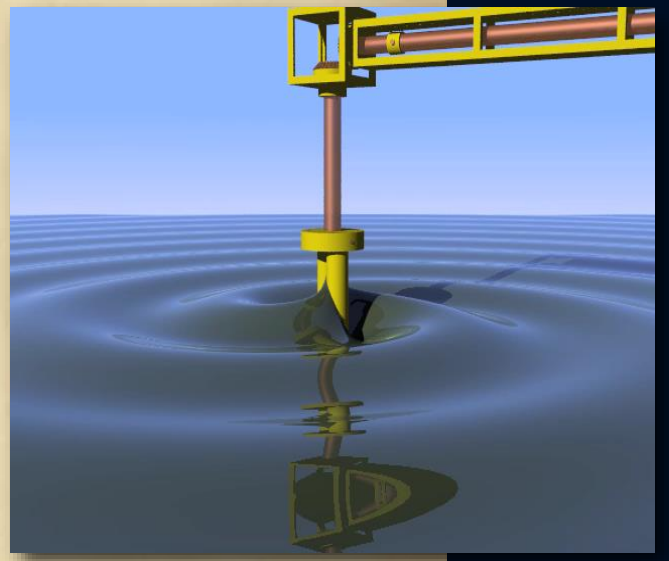
"1+log" slicing condition: $$f = 2/\alpha$$ where $f(\alpha) > 0$ and $K_0 := K(t = 0)$

"Gamma-Driver" shift condition:

$$\partial_t \beta^i - \beta^j \partial_j \beta^i = \frac{3}{4} B^i,$$

$$\partial_t B^i - \beta^j \partial_j B^i = \partial_t \tilde{\Gamma}^i - \beta^j \partial_j \tilde{\Gamma}^i - \eta B^i$$





Über Gravitationswellen.

Von A. Einstein.

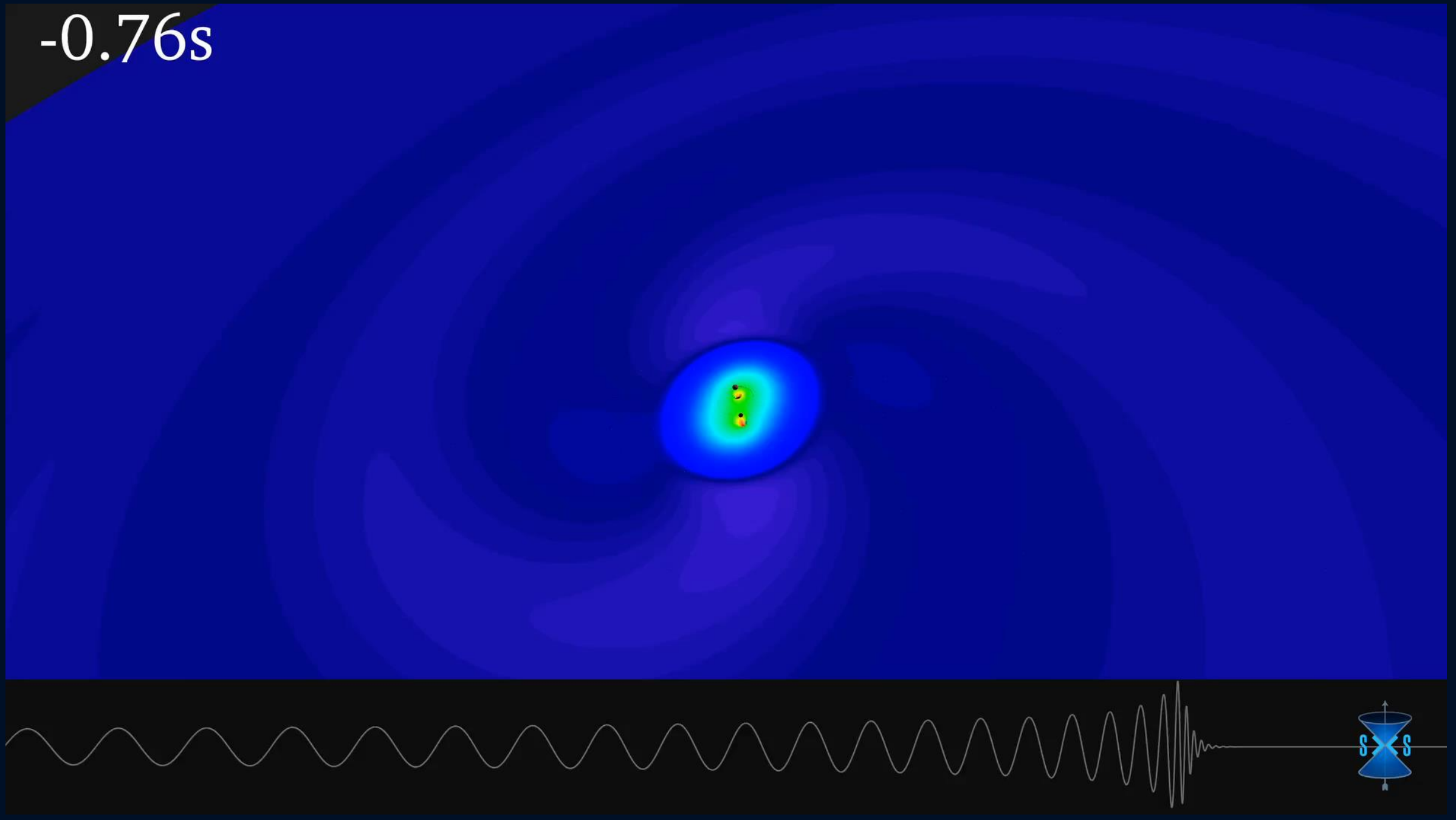
(Vorgelegt am 31. Januar 1918 [s. oben S. 79].)

Die wichtige Frage, wie die Ausbreitung der Gravitationsfelder erfolgt, ist schon vor anderthalb Jahren in einer Akademiearbeit von mir behandelt worden[1]. Da aber meine damalige Darstellung des Gegenstandes nicht genügend durchsichtig und außerdem durch einen bedauerlichen Rechenfehler verunstaltet ist, muß ich hier nochmals auf die Angelegenheit zurückkommen.

Einsteins erste Arbeit über Gravitationswellen

Sitzungsberichte der Königlich-Preußischen Akademie der Wissenschaften

Kollidierende Schwarze Löcher



100 Jahre später LIGO:
LIGO: **Laser Interferometer Gravitational-Wave Observatory**

PRL **116,** 061102 (2016)

Selected for a Viewpoint in *Physics*
PHYSICAL REVIEW LETTERS

week ending
12 FEBRUARY 2016

Observation of Gravitational Waves from a Binary Black Hole Merger

B. P. Abbott *et al.*[*]
(LIGO Scientific Collaboration and Virgo Collaboration)
(Received 21 January 2016; published 11 February 2016)

On September 14, 2015 at 09:50:45 UTC the two detectors of the Laser Interferometer Gravitational-Wave Observatory simultaneously observed a transient gravitational-wave signal. The signal sweeps upwards in frequency from 35 to 250 Hz with a peak gravitational-wave strain of 1.0×10^{-21}. It matches the waveform predicted by general relativity for the inspiral and merger of a pair of black holes and the ringdown of the resulting single black hole. The signal was observed with a matched-filter signal-to-noise ratio of 24 and a false alarm rate estimated to be less than 1 event per 203 000 years, equivalent to a significance greater than 5.1σ. The source lies at a luminosity distance of 410^{+160}_{-180} Mpc corresponding to a redshift $z = 0.09^{+0.03}_{-0.04}$. In the source frame, the initial black hole masses are $36^{+5}_{-4} M_\odot$ and $29^{+4}_{-4} M_\odot$, and the final black hole mass is $62^{+4}_{-4} M_\odot$, with $3.0^{+0.5}_{-0.5} M_\odot c^2$ radiated in gravitational waves. All uncertainties define 90% credible intervals. These observations demonstrate the existence of binary stellar-mass black hole systems. This is the first direct detection of gravitational waves and the first observation of a binary black hole merger.

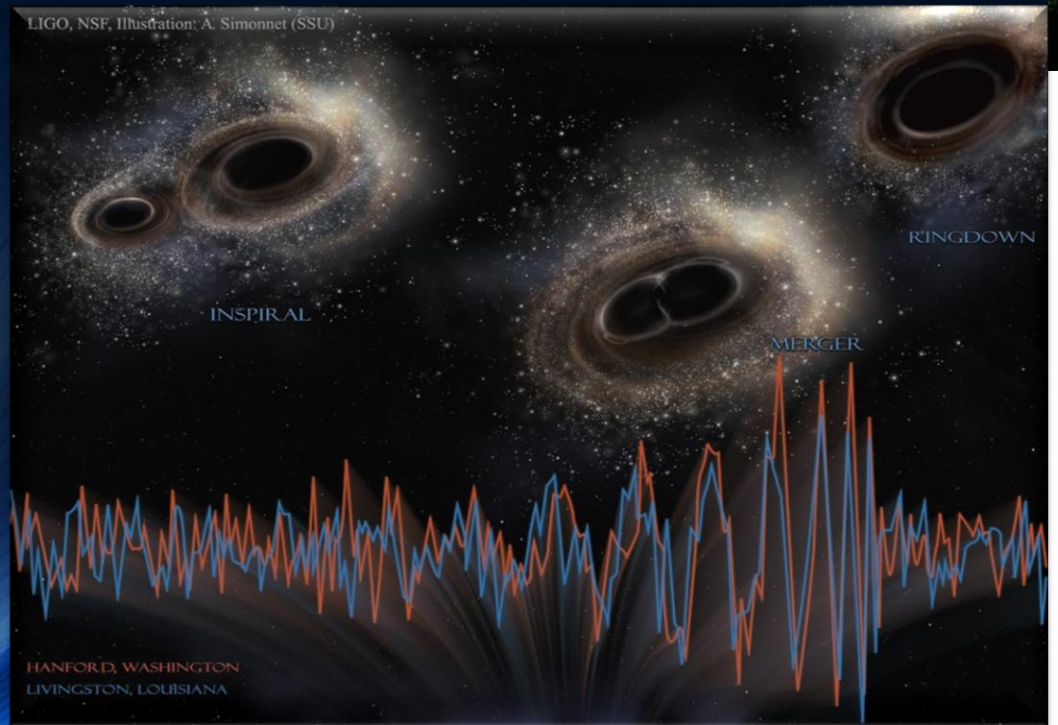
1. Direkter Nachweis von Gravitationswellen
Signalform: Verschmelzung von zwei **schwarzen** Löchern

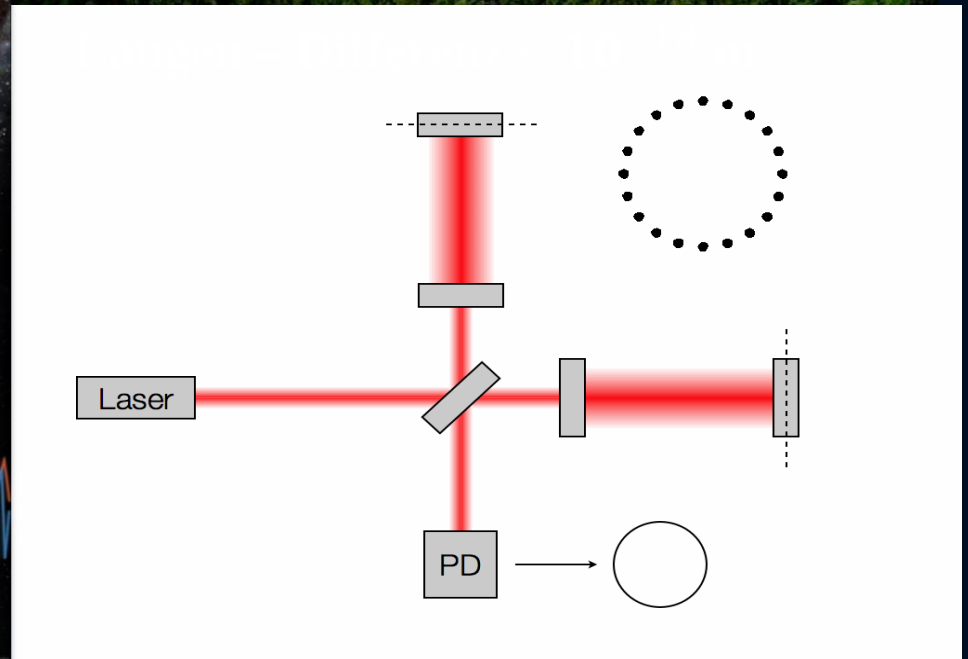
Gravitationswellen gefunden: LIGO!!!

Kollision zweier Schwarzer Löcher GW150914

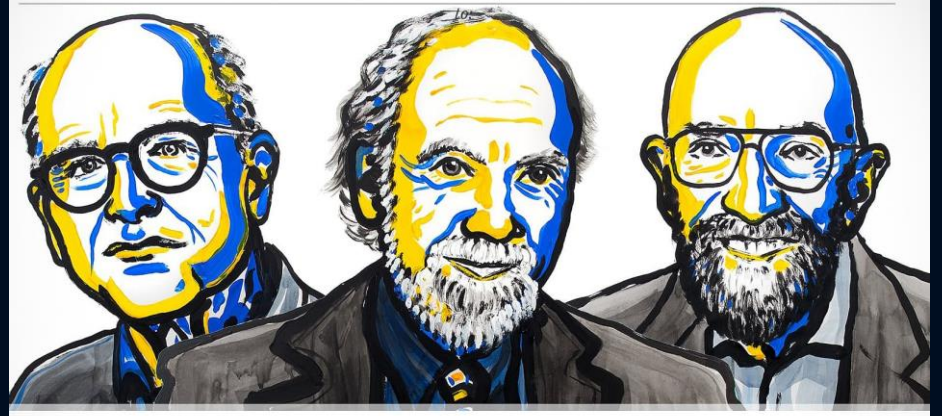
Massen: 36 & 29 Sonnenmassen

Abstand zur Erde 410 Mpc (1.34 Milliarden Lichtjahre)

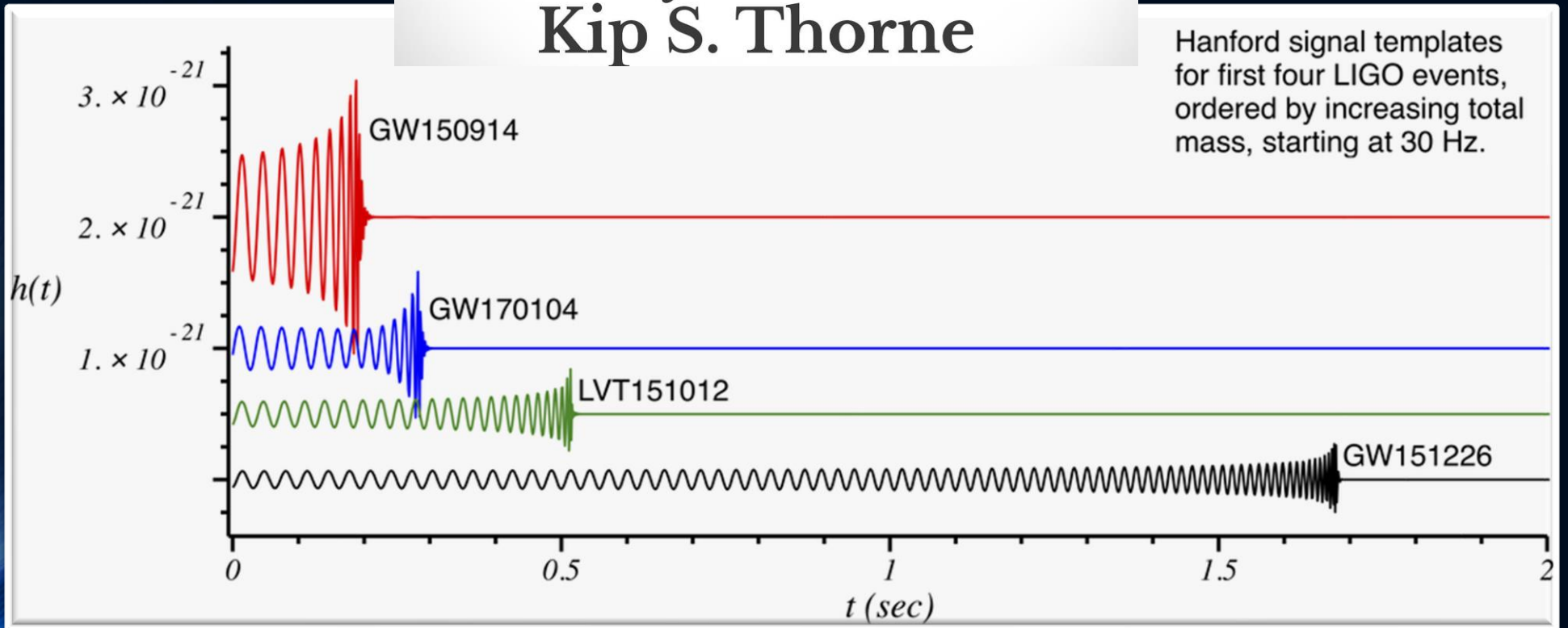




2017 NOBEL PRIZE IN PHYSICS

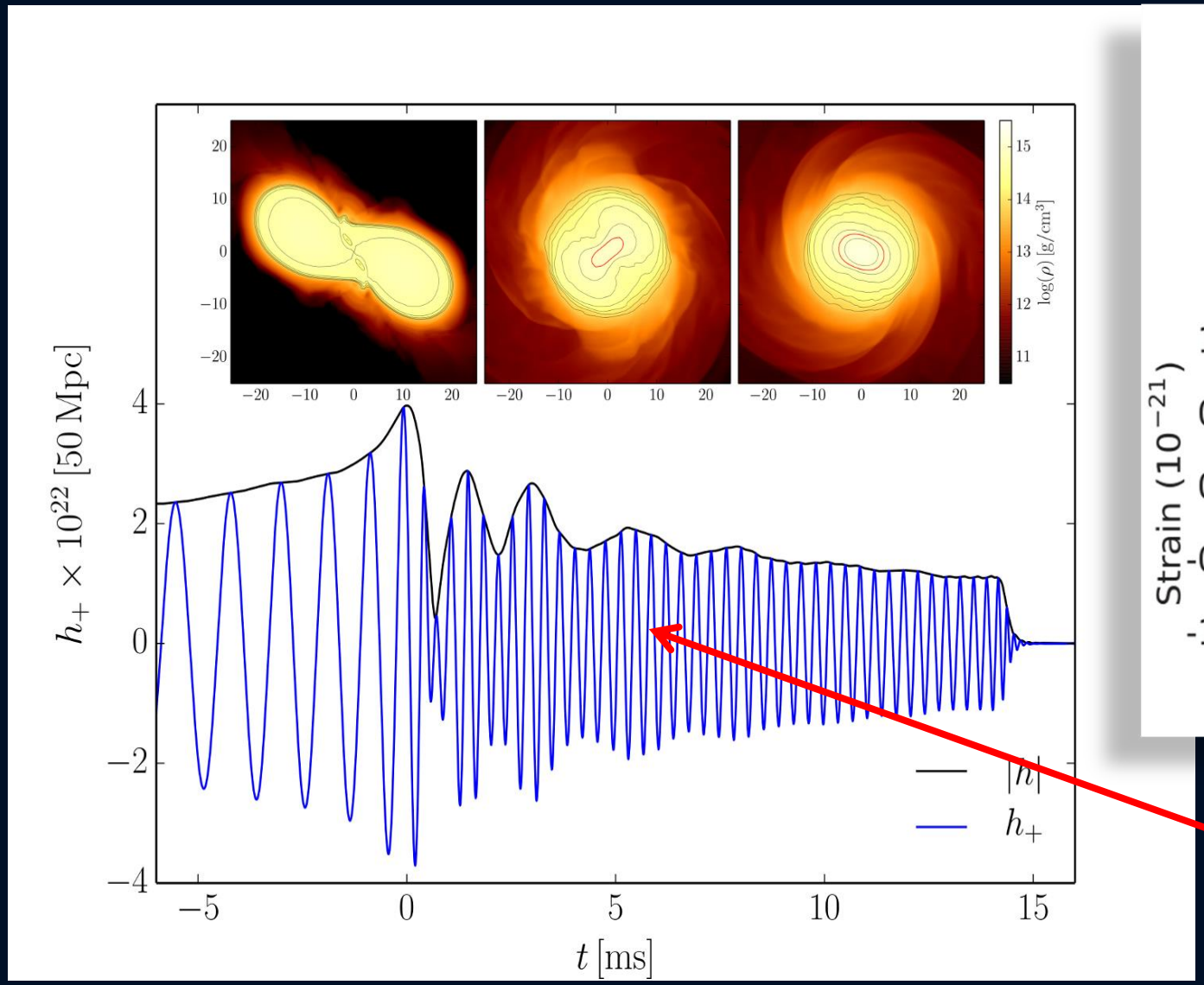


Rainer Weiss
Barry C. Barish
Kip S. Thorne

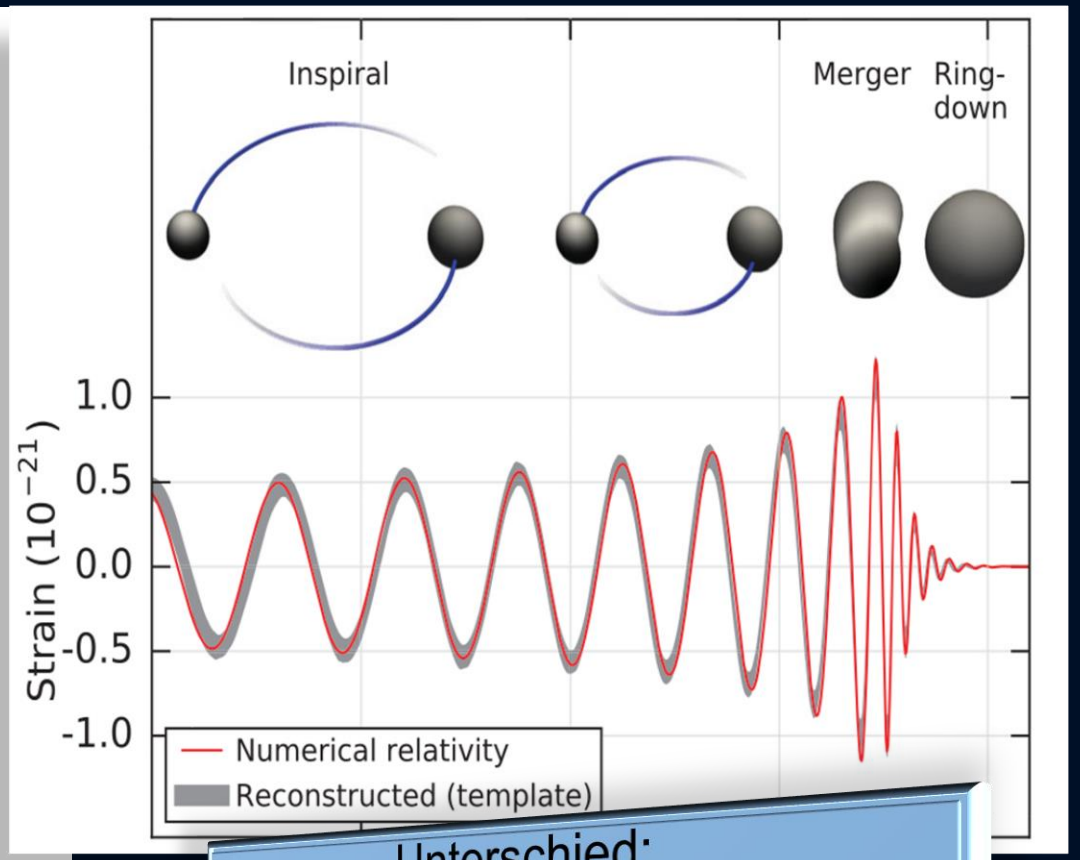


Gravitationswellen von Neutronenstern Kollisionen

Neutronenstern Kollision (Simulation)



Kollision zweier schwarzer Löcher

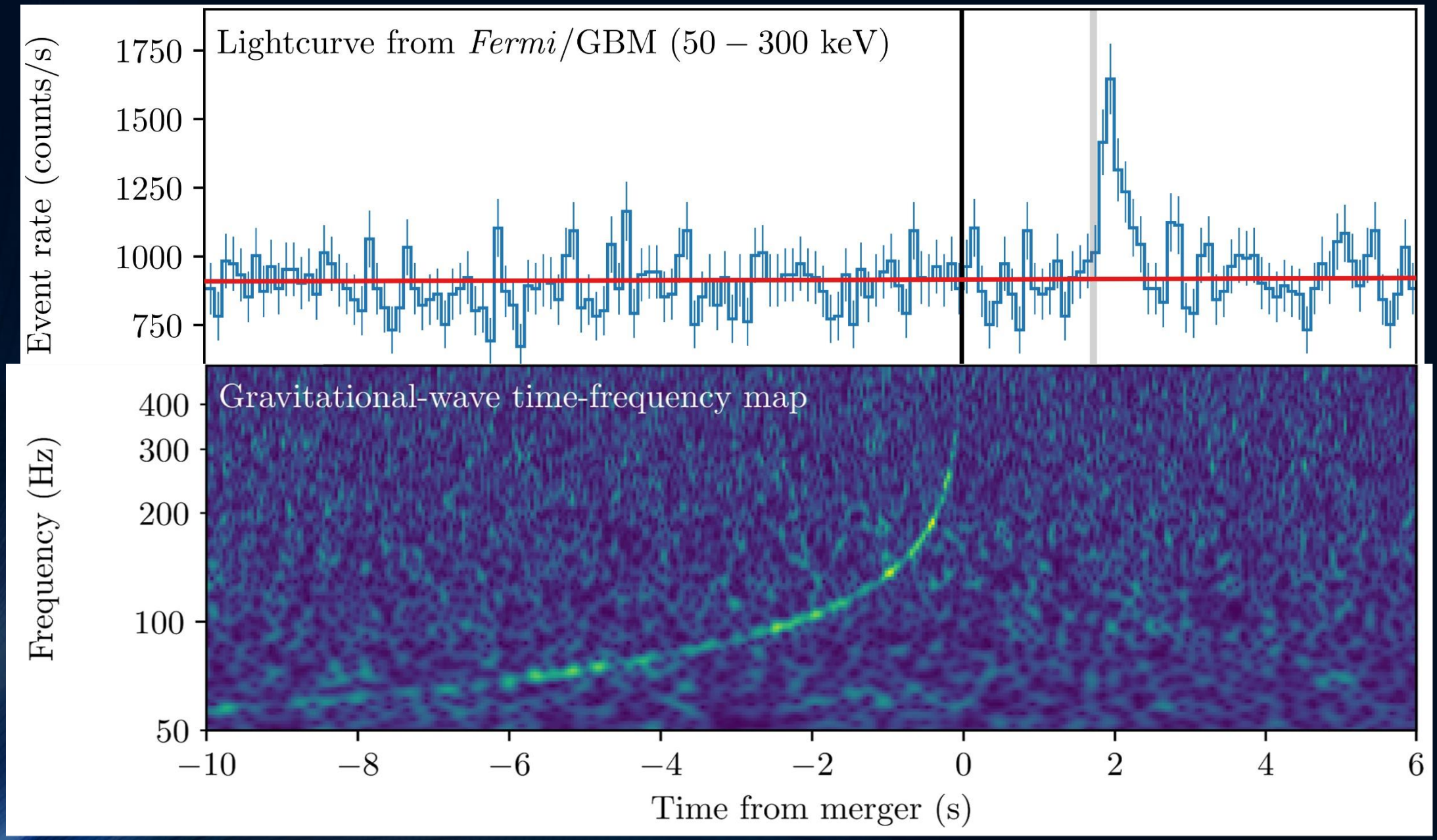


Unterschied:
Bei Neutronenstern Kollisionen gibt es meistens eine ***Post-Kollisionsphase***

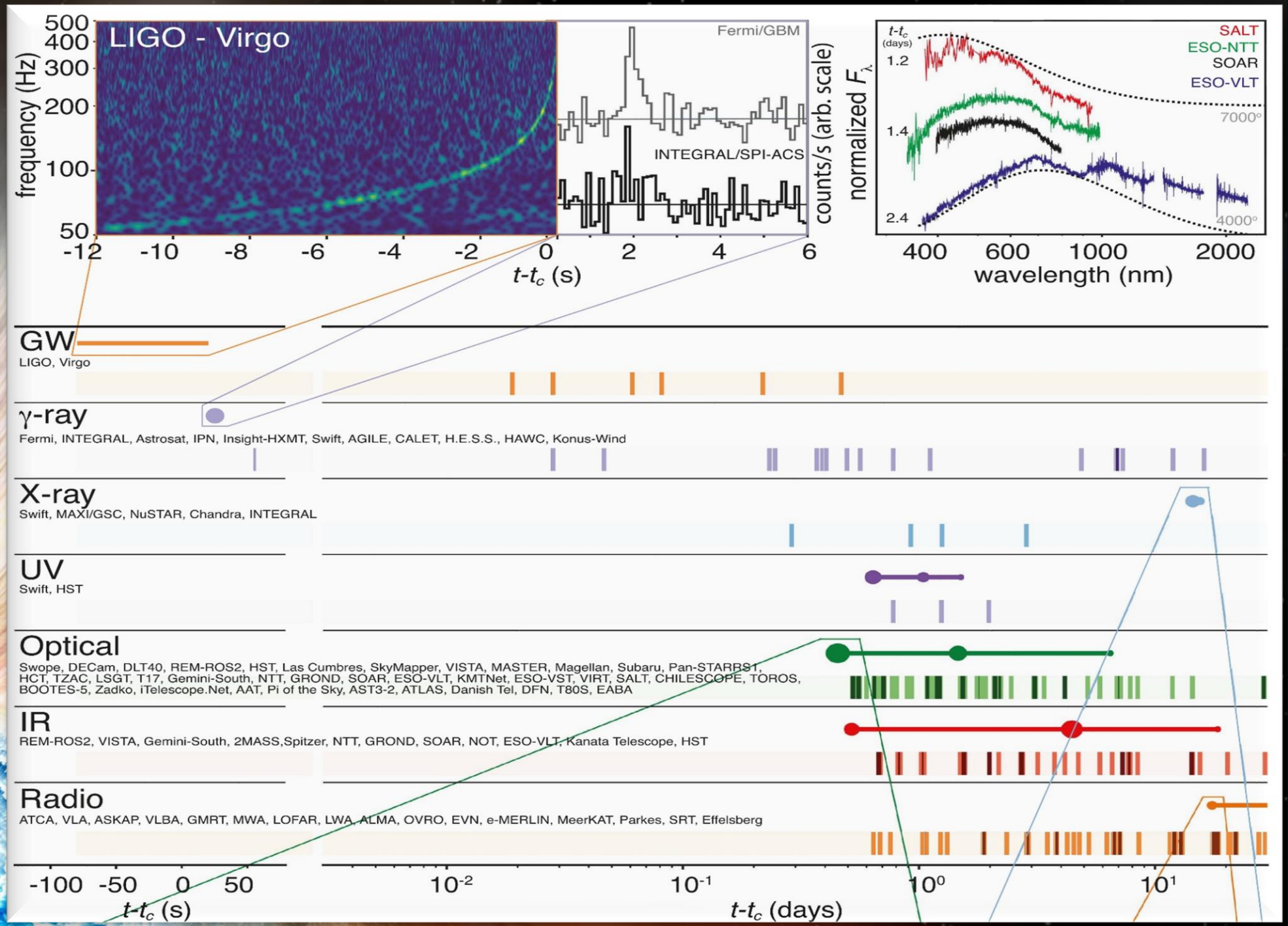
The long-awaited event GW170817

	Low-spin priors ($\lvert\chi\rvert \leq 0.05$)	High-spin priors ($\lvert\chi\rvert \leq 0.89$)
Primary mass m_1	1.36–1.60 $M_\odot$	1.36–2.26 $M_\odot$
Secondary mass m_2	1.17–1.36 $M_\odot$	0.86–1.36 $M_\odot$
Chirp mass $\mathcal{M}$	$1.188^{+0.004}_{-0.002} M_\odot$	$1.188^{+0.004}_{-0.002} M_\odot$
Mass ratio m_2/m_1	0.7–1.0	0.4–1.0
Total mass $m_{\rm tot}$	$2.74^{+0.04}_{-0.01} M_\odot$	$2.82^{+0.47}_{-0.09} M_\odot$
Radiated energy $E_{\rm rad}$	$> 0.025 M_\odot c^2$	$> 0.025 M_\odot c^2$
Luminosity distance $D_{\rm L}$	40^{+8}_{-14} Mpc	40^{+8}_{-14} Mpc
Viewing angle Θ	$\leq 55°$	$\leq 56°$
Using NGC 4993 location	$\leq 28°$	$\leq 28°$
Combined dimensionless tidal deformability $\tilde{\Lambda}$	≤ 800	≤ 700
Dimensionless tidal deformability $\Lambda(1.4 M_\odot)$	≤ 800	≤ 1400

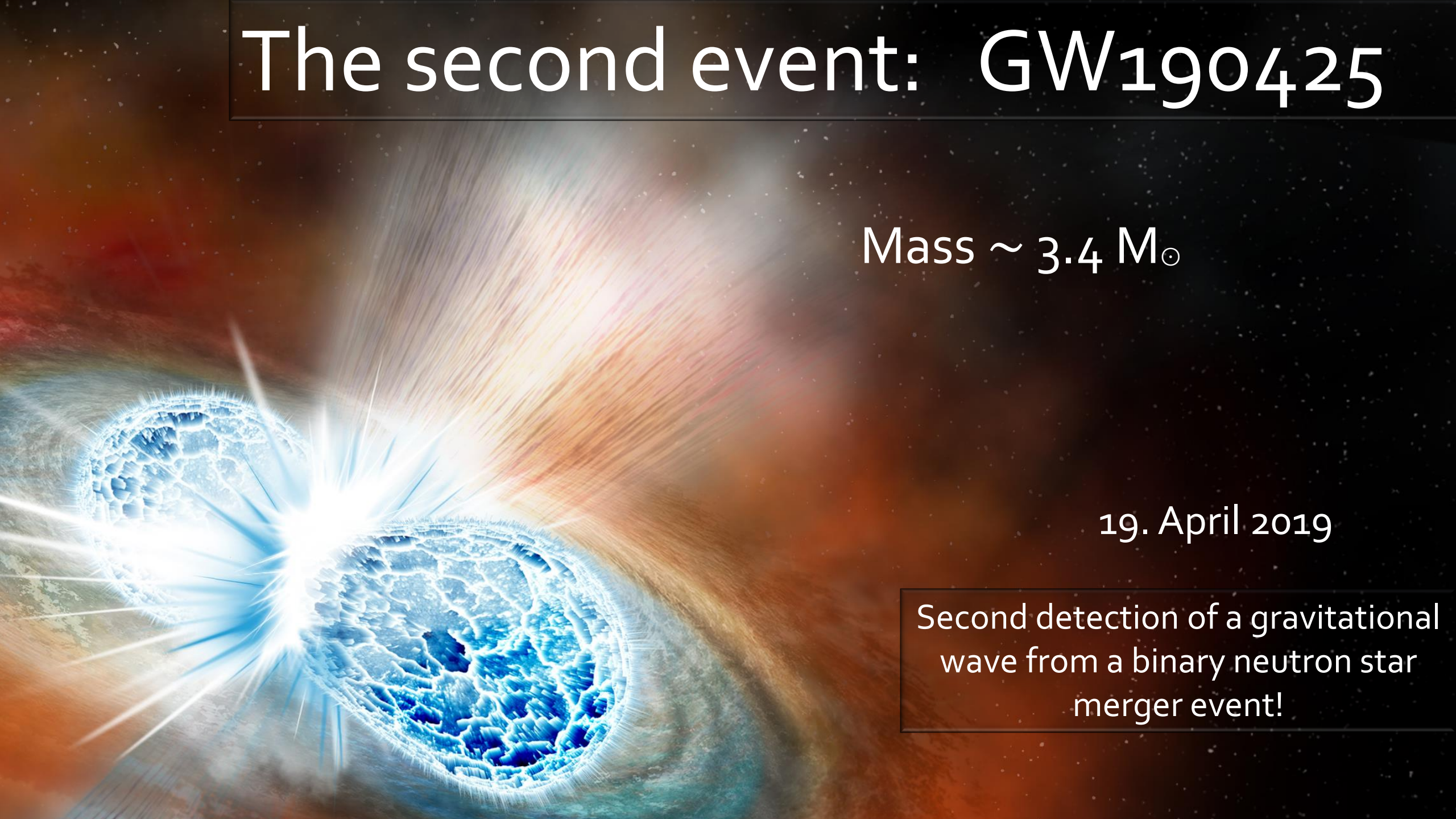
Gravitational Wave GW170817 and Gamma-Ray Emission GRB170817A



GW170817


LIGO - Virgo
frequency (Hz)
500
400
300
200
100
50
-12
-10
-8
-6
-4
-2
0
t-t_c (s)
Fermi/GBM
INTEGRAL/SPI-ACS
counts/s (arb. scale)
2
4
6
SALT
ESO-NTT
SOAR
ESO-VLT
t-t_c (days)
1.2
1.4
2.4
7000°
4000°
normalized F_λ
400
600
1000
2000
wavelength (nm)
GW
LIGO, Virgo
γ-ray
Fermi, INTEGRAL, Astrosat, IPN, Insight-HXMT, Swift, AGILE, CALET, H.E.S.S., HAWC, Konus-Wind
X-ray
Swift, MAXI/GSC, NuSTAR, Chandra, INTEGRAL
UV
Swift, HST
Optical
Swope, DECam, DLT40, REM-ROS2, HST, Las Cumbres, SkyMapper, VISTA, MASTER, Magellan, Subaru, Pan-STARRS1, HCT, TZAC, LSGT, T17, Gemini-South, NTT, GROND, SOAR, ESO-VLT, KMTNet, ESO-VST, VIRT, SALT, CHILESCOPE, TOROS, BOOTES-5, Zadko, iTelescope.Net, AAT, Pi of the Sky, AST3-2, ATLAS, Danish Tel, DFN, T80S, EABA
IR
REM-ROS2, VISTA, Gemini-South, 2MASS,Spitzer, NTT, GROND, SOAR, NOT, ESO-VLT, Kanata Telescope, HST
Radio
ATCA, VLA, ASKAP, VLBA, GMRT, MWA, LOFAR, LWA, ALMA, OVRO, EVN, e-MERLIN, MeerKAT, Parkes, SRT, Effelsberg
-100
-50
0
50
t-t_c (s)
10^{-2}
10^{-1}
10^{0}
10^{1}
t-t_c (days)

The second event: GW190425



Mass ~ 3.4 $M_\odot$

19. April 2019

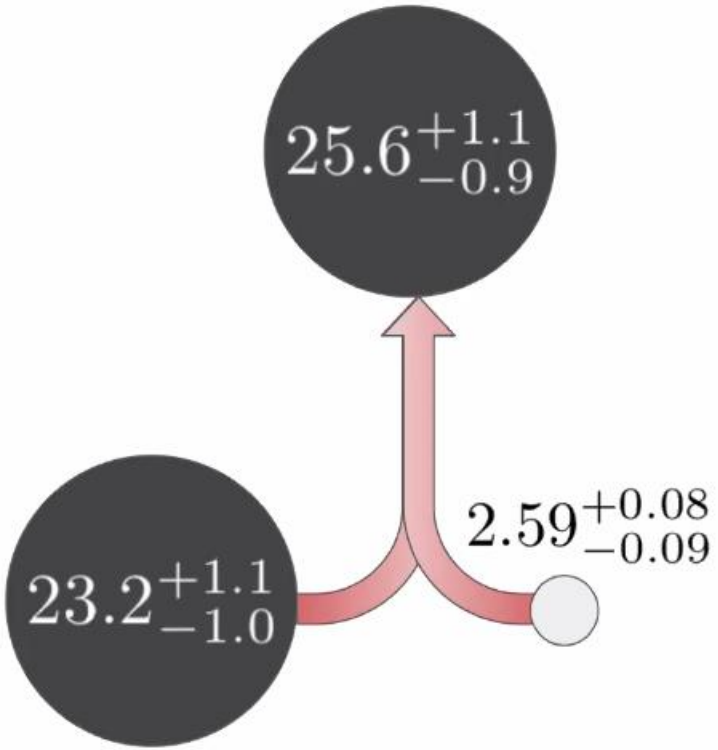
Second detection of a gravitational wave from a binary neutron star merger event!

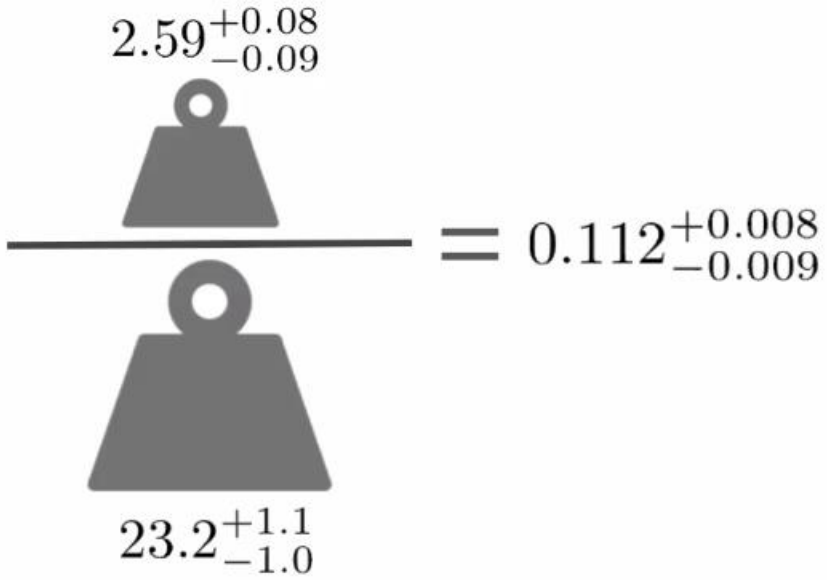
GW190814

The third event ???

14. August 2019

GW190814 - Source frame masses

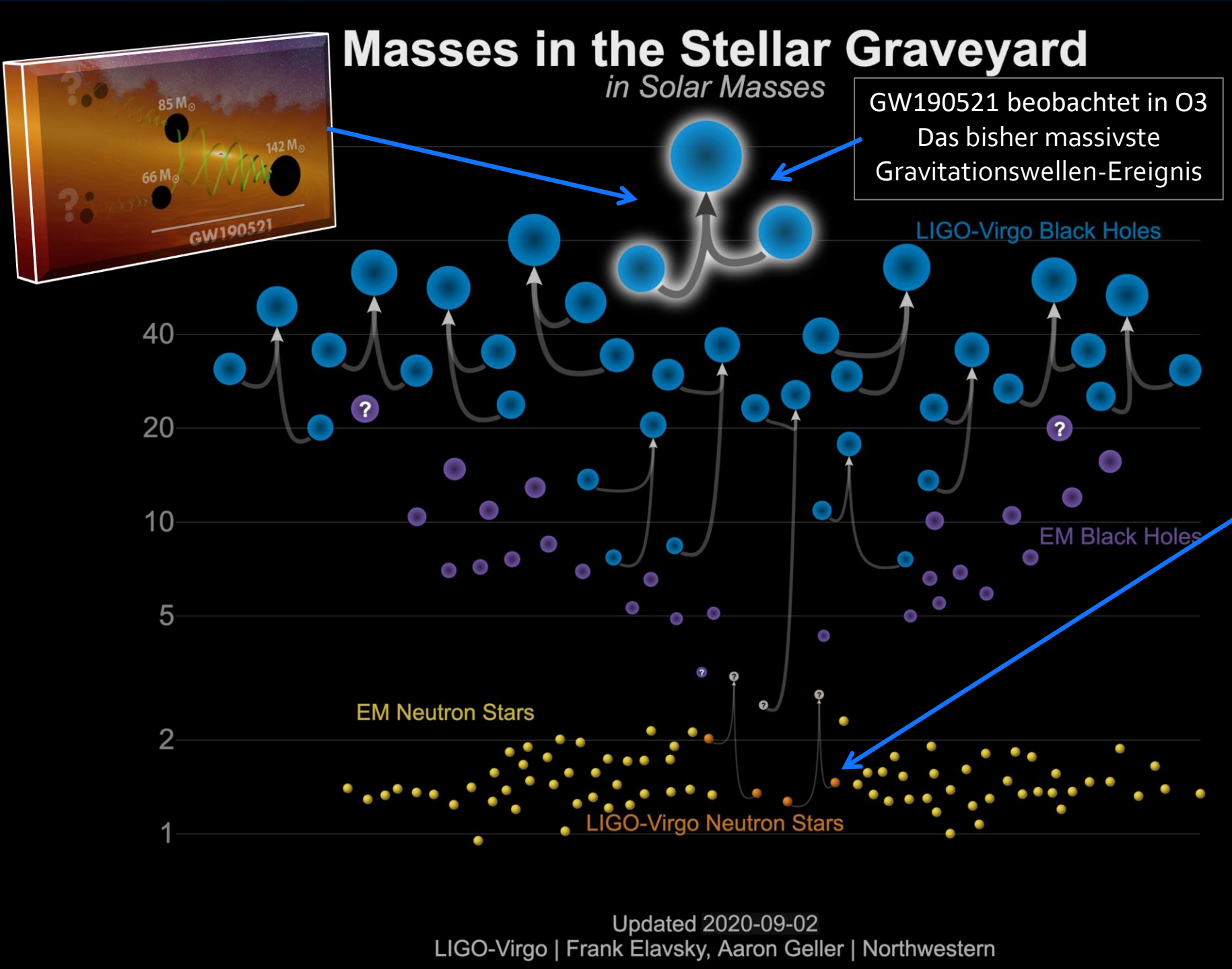


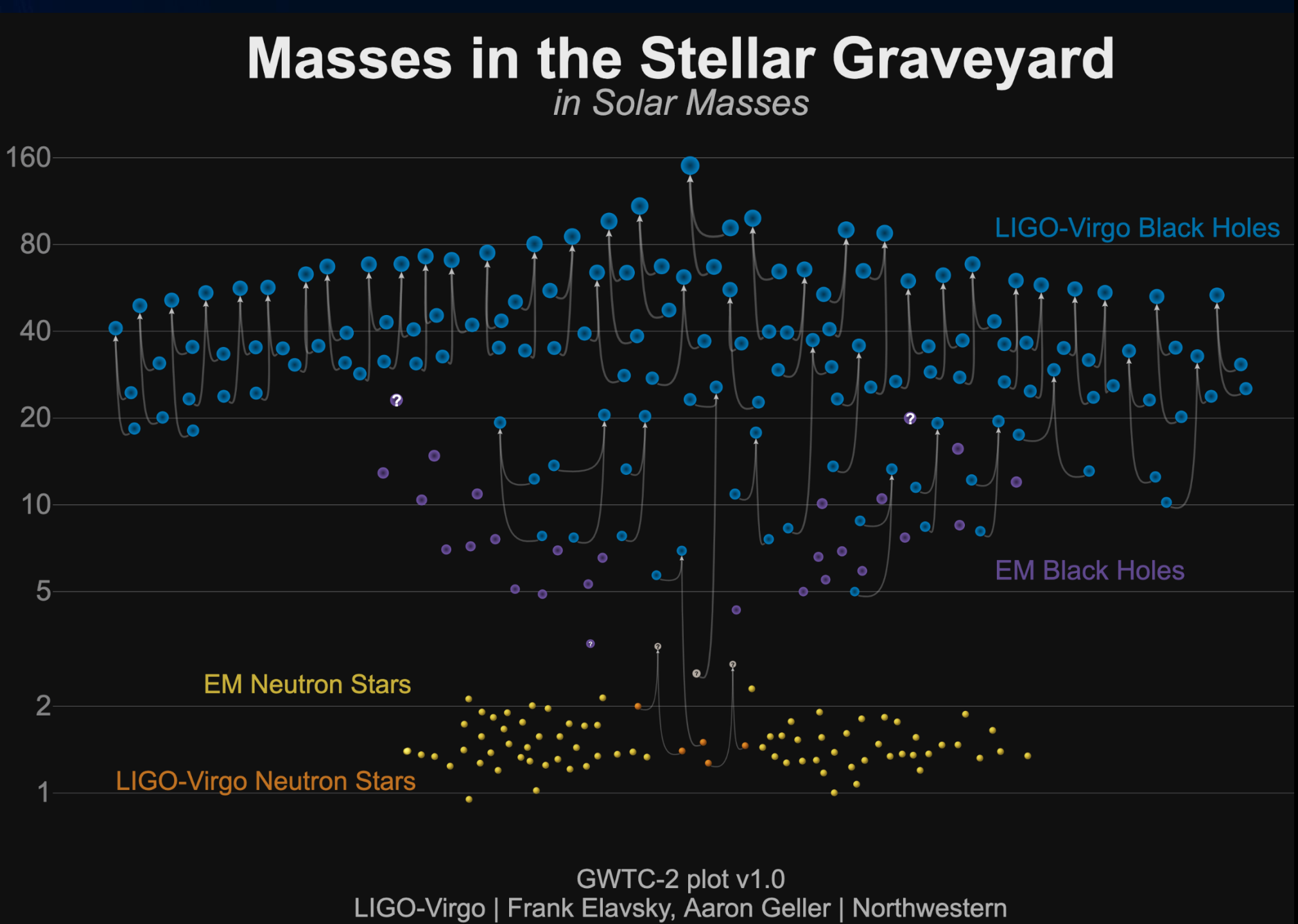


8

Detektierte Gravitationswellen

In den ersten beiden Beobachtungsläufen (O1+O2) konnten 11 Gravitationswellen detektiert werden, wobei einer dieser Gravitationswellen (GW170817) durch die Kollision zweier Neutronensterne verursacht wurde welche sich vor ungefähr 130 Millionen Jahren ereignete.





Am 28. Oktober 2020 gaben die LIGO Scientific Collaboration und die Virgo Collaboration die Ergebnisse ihrer Suche nach Gravitationswellen aus kollidierenden kompakten Objekten in der ersten Hälfte ihres dritten Beobachtungslaufs (O3a: vom 1. 04.2919 bis 1.10.2019) bekannt. 39 Ereignisse wurden hier gemeldet. Zusammen mit den 11 Ereignissen im ersten Gravitationswellen-Katalog (GWTC-1), erhöht sich die Gesamtzahl der gemeldeten LIGO / Virgo-Gravitationswellen-Ereignissen auf 50.

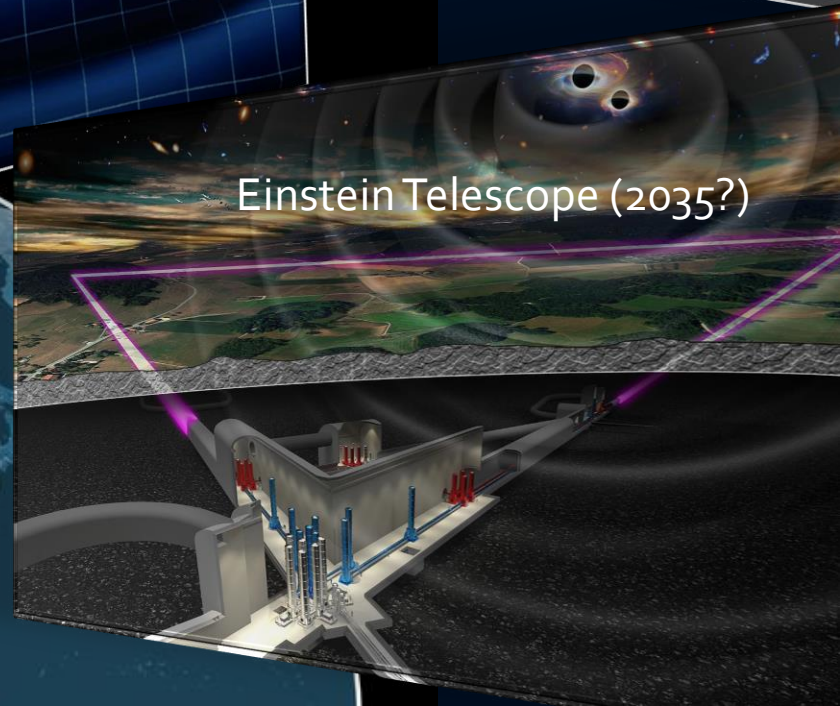
Im Beobachtungszeitraum O3 gefundene GW

Laser Interferometer Space Antenna
LISA (2034)

Cosmic Explorer (2035?)

The next observing runs (O4,O5,..)

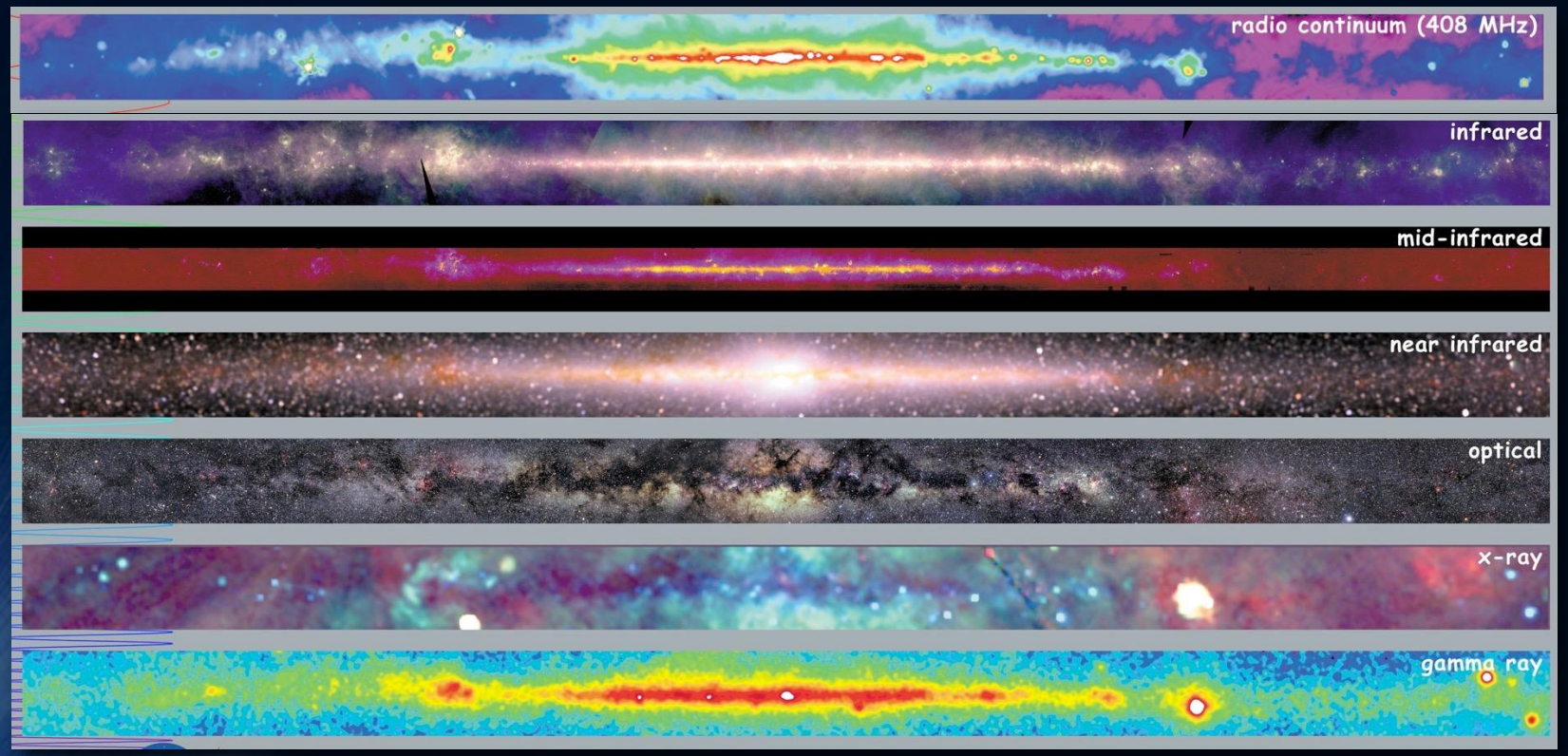
Einstein Telescope (2035?)



Die neue Art unser Universum zu betrachten

Lange Zeit über war das Studium von astrophysikalischen Vorgängen auf den mit den Augen sichtbaren Bereich limitiert und optische Teleskope entwickelten sich erst ab dem 16. Jahrhundert. Die Wahrnehmung des Universums in den anderen Frequenzbereichen der elektromagnetischen Strahlung wurde durch die Radio-, Infrarot- und Röntgen-Teleskope möglich und entwickelte sich erst im 20. Jahrhundert.

GSFC/NASA



Radio

w-IR

m-IR

n-IR

Optisch

Röntgen

Gamma

Gravitations-wellen

Erkenntnisse mittels elektromagnetische Strahlung

Es ist so als ob die Menschheit eine neue wundersame Brille hat, ein neues Sinnesorgan, mit welchem sie nun, zuvor unbeobachtbare Ereignisse in unserem Universum, wahrnehmen kann.