Marine Data Literacy Course - Practical 1
Model and Satellite CMEMS Sea Surface Temperature data

Adam Gauci and Aldo Drago

Contents:

Introduction

This instruction sheet is meant to describe and list the structured activities, instructions and expected outputs for the practical exercise. It also serves to support the students to follow and execute the various steps of the exercise.

It is suggested that these instructions are kept open and available during the practical session such that instructions and pieces of code can be used directly without rewriting.

The students will asked to answer four MC questions which are based on outputs from the practical session. The tutors delivering the practical will guide you to prepare your answers at the appropriate stages of your practical, so that you will then submit your answers at the end of the session.

Targets of the practical

The Copernicus Marine Environment Monitoring Service (CMEMS) is an EU information service based on satellite earth observations and in-situ data. CMEMS provides state-of-the-art analyses and forecasts of oceanographic parameters which offer an unprecedented capability to observe, understand, and anticipate marine environment events. CMEMS offers unique access to

oceanographic products through an online catalogue. In this practical the students will learn how to:

- Log in the Marine Copernicus Portal.
- Download daily averaged sea surface temperature (SST) data from a Mediterranean operational model that is available for ten consecutive forecast days.
- Download satellite observed SST data (L4) generated over the same domain and ten-day period. Load and visualise the model and satellite netcdf files in Panoply.
- Visualise how model sea temperature varies with depth and time.
- Identify a grid cell in the model and satellite datasets, and extract the SST time series over the ten days being considered.
- Regrid raster data to be able to compare two datasets.
- Generate a scatter plot to visualize the correlation between the model and the satellite. Compute the Mean Square Error between the model and satellite SST.

Before the practical, the students are expected to:

1. Download and install Panoply from https://www.giss.nasa.gov/tools/panoply/download/; this is the application that will be used for plotting
2. Download and install Anaconda Individual Edition from https://www.anaconda.com/products/individual; to run Python code on your local machine using Jupyter Notebook OR use a Google Account to log into Colaboratory by visiting https://colab.research.google.com to run the code on as remote server.
3. Register for a CMEMS account from: https://marine.copernicus.eu from where to choose and download the data for this exercise
4. Download a copy of the files that were specifically used for this demonstartion from **here**.

IMPORTANT: These installations need to be done ahead of the practical session so that their functionality can be tested BEFORE the session.

Datasets used in the practical

In this practical, files from the 'Mediterranean Sea High Resolution and Ultra High Resolution Sea Surface Temperature Analysis' model as well as from the 'Mediterranean Sea High Resolution and Ultra High Resolution Sea Surface Temperature Analysis' satellite datasets, will be used. Both products are downloadable from the Marine Copernicus Portal (CMEMS).

Mediterranean Sea High Resolution and Ultra High Resolution Sea Surface Temperature Analysis

The physical component of the Mediterranean Forecasting System (Med-Currents) is a coupled hydrodynamic-wave model implemented over the whole Mediterranean Basin including tides. The model horizontal grid resolution is 1/24˚ (ca. 4 km) and has 141 unevenly spaced vertical levels. The hydrodynamics are supplied by the Nucleous for European Modelling of the Ocean (NEMO v3.6) while the wave component is provided by Wave Watch-III; the model solutions are corrected by a variational data assimilation scheme (3DVAR) of temperature and salinity vertical profiles and along track satellite Sea Level Anomaly observations.

Filename: `med-cmcc-tem-an-fc-h_1634968902280.nc`
Product type: `Forecast`
Spatial resolution: `0.042° × 0.042°`

Vertical coverage: `from 1.01824 down to 1005.14 (74 levels)`
Time period: `20/10/2021 00:30 to 22/10/2021 23:30 (72 time steps)`

Mediterranean Sea High Resolution and Ultra High Resolution Sea Surface Temperature Analysis

For the Mediterranean Sea (MED), the CNR MED Sea Surface Temperature (SST) processing chain provides daily gap-free (L4) maps at high (HR 0.0625°) and ultra-high (UHR 0.01°) spatial resolution over the Mediterranean Sea. Remotely-sensed L4 SST datasets are operationally produced and distributed in near-real time by the Consiglio Nazionale delle Ricerche - Gruppo di Oceanografia da Satellite (CNR-GOS). These SST products are based on the nighttime images collected by the infrared sensors mounted on different satellite platforms, and cover the Southern European Seas. The CNR-GOS processing chain includes several modules, from the data extraction and preliminary quality control, to cloudy pixel removal and satellite images collating/merging. A two-step algorithm finally allows to interpolate SST data at high (HR 0.0625°) and ultra-high (UHR 0.01°) spatial resolution, applying statistical techniques. These L4 data are also used to estimate the SST anomaly with respect to a pentad climatology.

Filename: `SST_MED_SST_L4_NRT_OBSERVATIONS_010_004_c_V2_1634969733519.nc`
Product type: `Satellite (L4)`
Spatial resolution: `0.042° × 0.042°`
Vertical coverage: `surface (1 level)`
Time period: `20/10/2021 00:00 to 22/10/2021 23:00 (3 time steps)`

Initialising the Python Environment

Installing the required libraries that will eventually be used by this program.

In [1]:
```
pip install netCDF4
```

```
Requirement already satisfied: netCDF4 in /Users/adamgauci/opt/anaconda3/lib/python
3.8/site-packages (1.5.8)
Requirement already satisfied: cftime in /Users/adamgauci/opt/anaconda3/lib/python3.
8/site-packages (from netCDF4) (1.5.1.1)
Requirement already satisfied: numpy>=1.9 in /Users/adamgauci/opt/anaconda3/lib/pyth
on3.8/site-packages (from netCDF4) (1.20.1)
Note: you may need to restart the kernel to use updated packages.
```

Importing the required libraries in Python.

In [2]:
```
import netCDF4 as nc
import matplotlib.pyplot as plt
import datetime
import time
import IPython.display as display
import scipy.interpolate as interp
import numpy as np
```

Processing model data

Reading the model netcdf file

Defining the filename and importing the netcdf dataset.

In [3]:
```
modelFilename = 'Data/med-cmcc-tem-an-fc-h_1634968902280.nc'
modelDataSet = nc.Dataset(modelFilename)
```

Printing the metadata of the netcdf file and of the 'thetao' and 'time' varibales to check dimensions, units, and see how these are defined.

In [4]:
```
print(modelDataSet)
```

```
<class 'netCDF4._netCDF4.Dataset'>
root group (NETCDF3_CLASSIC data model, file format NETCDF3):
    title: Potential Temperature (3D) - Hourly Mean
    FROM_ORIGINAL_FILE__field_type: hourly_mean_centered_at_time_field
    source: MFS EAS6
    institution: Centro Euro-Mediterraneo sui Cambiamenti Climatici - CMCC, Italy
    contact: servicedesk.cmems@mercator-ocean.eu
    references: Clementi, E., Aydogdu, A., Goglio, A. C., Pistoia, J., Escudier, R.,
Drudi, M., Grandi, A., Mariani, A., Lyubartsev, V., Lecci, R., Creti, S., Coppini,
G., Masina, S., & Pinardi, N. (2021). Mediterranean Sea Analysis and Forecast (CMEMS
MED-Currents, EAS6 system) (Version 1) [Data set]. Copernicus Monitoring Environment
Marine Service (CMEMS). https://doi.org/10.25423/CMCC/MEDSEA_ANALYSISFORECAST_PHY_00
6_013_EAS6
    comment: Please check in CMEMS catalogue the INFO section for product MEDSEA_ANA
LYSISFORECAST_PHY_006_013 - http://marine.copernicus.eu
    Conventions: CF-1.0
    bulletin_date: 20211022
    bulletin_type: forecast
    _CoordSysBuilder: ucar.nc2.dataset.conv.CF1Convention
    history: Data extracted from dataset http://localhost:8080/thredds/dodsC/med-cmc
c-tem-an-fc-h
    dimensions(sizes): time(72), depth(74), lat(168), lon(193)
    variables(dimensions): float32 depth(depth), float32 thetao(time, depth, lat, lo
n), float32 lon(lon), float64 time(time), float32 bottomT(time, lat, lon), float32 l
at(lat)
    groups:
```

In [5]:
```
print (modelDataSet['thetao'])
```

```
<class 'netCDF4._netCDF4.Variable'>
float32 thetao(time, depth, lat, lon)
    _FillValue: 1e+20
    units: degrees_C
    standard_name: sea_water_potential_temperature
    long_name: potential temperature
    coordinates: time depth lat lon
    missing_value: 1e+20
    _ChunkSizes: [  1  29  76 258]
unlimited dimensions:
current shape = (72, 74, 168, 193)
filling on
```

In [6]:
```
print(modelDataSet['time'])
```

```
<class 'netCDF4._netCDF4.Variable'>
float64 time(time)
    units: minutes since 1900-01-01 00:00:00
    standard_name: time
    long_name: time
    axis: T
    calendar: standard
    _ChunkSizes: 24
```

```
    _CoordinateAxisType: Time
    valid_min: 64061310.0
    valid_max: 64065570.0
unlimited dimensions:
current shape = (72,)
filling on, default _FillValue of 9.969209968386869e+36 used
```

Reading the variables from the model nc file.

In [7]:
```
modelLon = modelDataSet['lon']
modelLat = modelDataSet['lat']
modelDepth = modelDataSet['depth']
modelTemperature = modelDataSet['thetao']
modelTime = modelDataSet['time']
```

Processing single frames

Extracting temperature values across all longitudes and all latitudes, for the first time frame (out of 72), and the first depth level (out of 74).

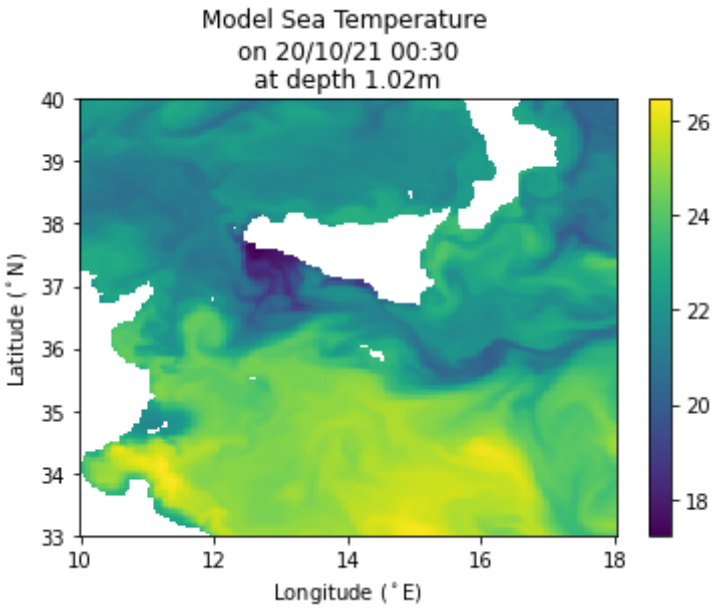
In [8]:
```
modelTemp1 = modelTemperature[0,0,:,:];
```

Calculating the time corresponding to this frame.

In [9]:
```
modelBaseDate =  datetime.datetime(1900, 1, 1, 0, 0, 0);

timeMins = float(modelTime[0])
modelTimeDT1 = modelBaseDate + datetime.timedelta(minutes = timeMins)
```

Plotting the extracted temperature data (modelTemp1).

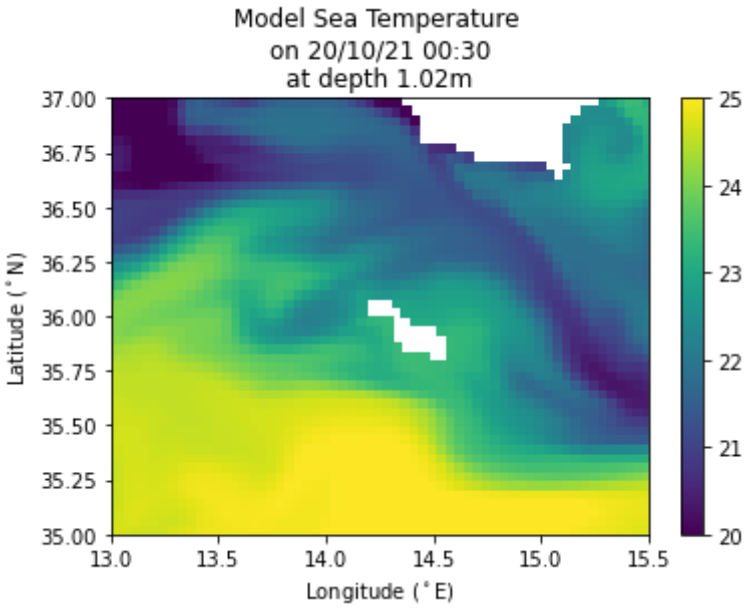
In [10]:
```
plt.pcolor(modelLon, modelLat, modelTemp1, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Model Sea Temperature \non " +  modelTimeDT1.strftime("%d/%m/%y %H:%M") +
plt.colorbar();
```



Plotting the extracted temperature data (modelTemp1), zooming on Malta, and adjusting the colorbar limits accordingly.

In [11]:

```
plt.pcolor(modelLon, modelLat, modelTemp1, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Model Sea Temperature \non " +  modelTimeDT1.strftime("%d/%m/%y %H:%M") +
plt.axis([13, 15.5, 35, 37]);
plt.colorbar();
plt.clim([20, 25]);
```

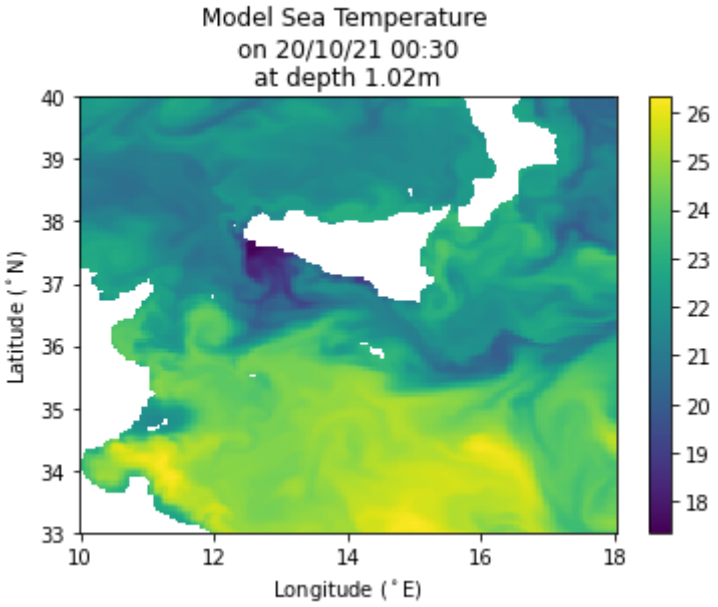


Exercise 1 (replace ?? with the correct terms):

- Extract temperature values across all longitudes and all latitudes, for the 24th time frame and the first depth level.
- Calculate the time corresponding to this frame.
- Plot the extracted temperature data (modelTemp2).

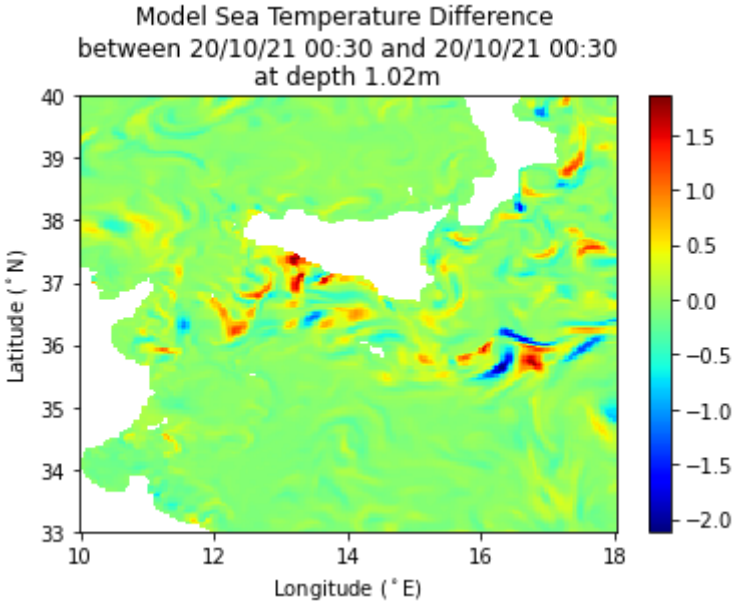
In [12]:

```
modelTemp2 = modelTemperature[24,0,:,:];

timeMins = float(modelTime[0])
modelTimeDT2 = modelBaseDate + datetime.timedelta(minutes = timeMins)

plt.pcolor(modelLon, modelLat, modelTemp2, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Model Sea Temperature \non " +  modelTimeDT2.strftime("%d/%m/%y %H:%M") +
plt.colorbar();
```



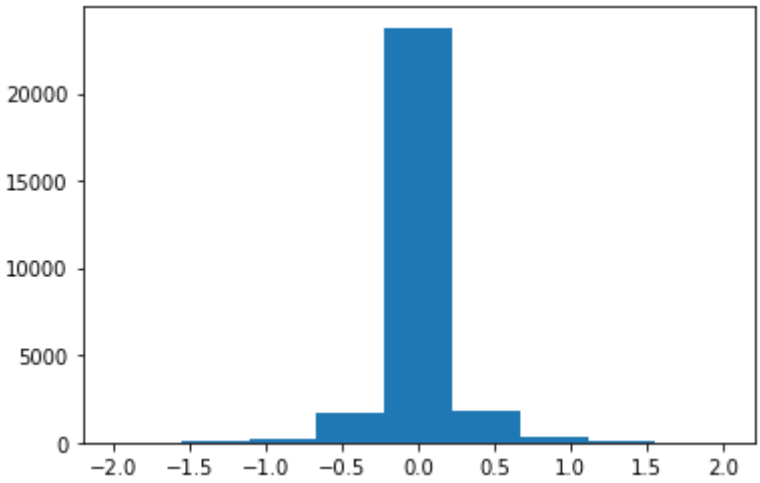
Compute and plot difference in tempeature between the two days.

In [13]:
```
modelDiff = modelTemp2 - modelTemp1;

plt.pcolor(modelLon, modelLat, modelDiff, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Model Sea Temperature Difference \nbetween " +  modelTimeDT1.strftime("%d
plt.colorbar();
plt.set_cmap('jet')
```



Generate a histogram of the residuals between -2 and 2, and use 10 bins.

In [14]:
```
plt.hist(modelDiff.flatten(), np.linspace(-2,2,10));
```



Quiz Question 1:
The majority of sea temperature values between the two days are:

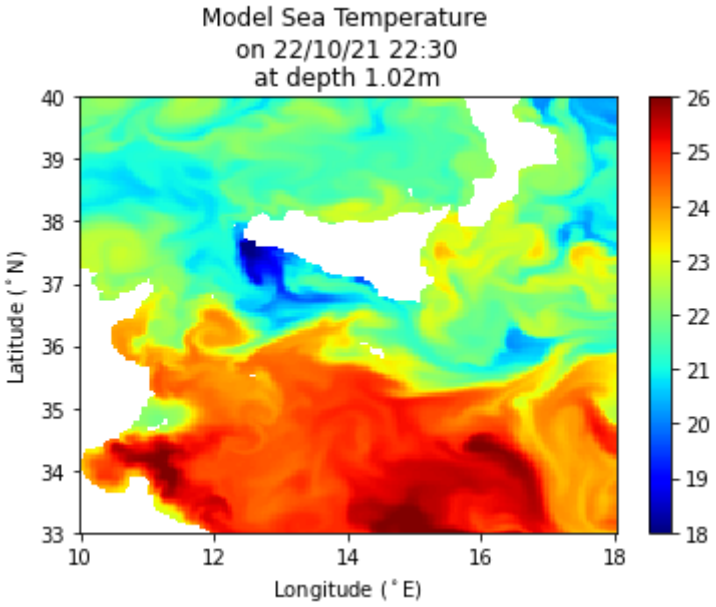
1. The same
2. Colder in day 1 than in day 2
3. Colder in day 2 than in day 1
4. This histogram does not give this information

Processing frames across time

Looping over all time frames (but showing every 5th frame).

In [15]:

```
nTimeFrames = modelTemperature.shape[0];

for t in range(0, (nTimeFrames-1), 5):
    d = 0;

    timeMins = float(modelTime[t])
    modelTimeCurr = modelBaseDate + datetime.timedelta(minutes = timeMins)

    modelTemp = modelTemperature[t,d,:,:];

    plt.clf()
    plt.pcolor(modelLon, modelLat, modelTemp, shading='auto');
    plt.ylabel("Latitude ($^\circ$N)");
    plt.xlabel("Longitude ($^\circ$E)");
    plt.title("Model Sea Temperature \non " +  modelTimeCurr.strftime("%d/%m/%y %H:%
    plt.colorbar();
    plt.clim([18, 26]);
    display.display(plt.gcf())
    display.clear_output(wait=True)
    #time.sleep(0.25)
```

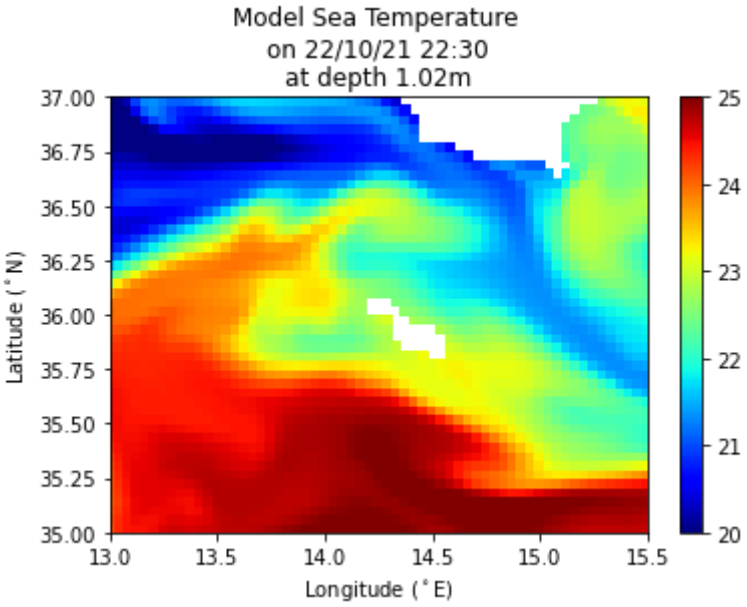


Exercise 2 (replace ?? with the correct terms):

- Loop over all time frames (but show every 5th frame).
- Set the colour bar limits between 18 and 26 degC.
- Zoom in the region between 13 degE and 15.5 degE in longitude, and 35 degN and 37 degN in latitude.

In [16]:

```
nTimeFrames = modelTemperature.shape[0];

for t in range(0, (nTimeFrames-1), 5):
    d = 0;

    timeMins = float(modelTime[t])
    modelTimeCurr = modelBaseDate + datetime.timedelta(minutes = timeMins)

    modelTemp = modelTemperature[t,d,:,:];

    plt.clf()
    plt.pcolor(modelLon, modelLat, modelTemp, shading='auto');
    plt.ylabel("Latitude ($^\circ$N)");
    plt.xlabel("Longitude ($^\circ$E)");
    plt.title("Model Sea Temperature \non " +  modelTimeCurr.strftime("%d/%m/%y %H:%
    plt.colorbar();
    plt.clim([20, 25]);
    plt.axis([13, 15.5, 35, 37]);
    display.display(plt.gcf())
    display.clear_output(wait=True)
    #time.sleep(0.25)
```

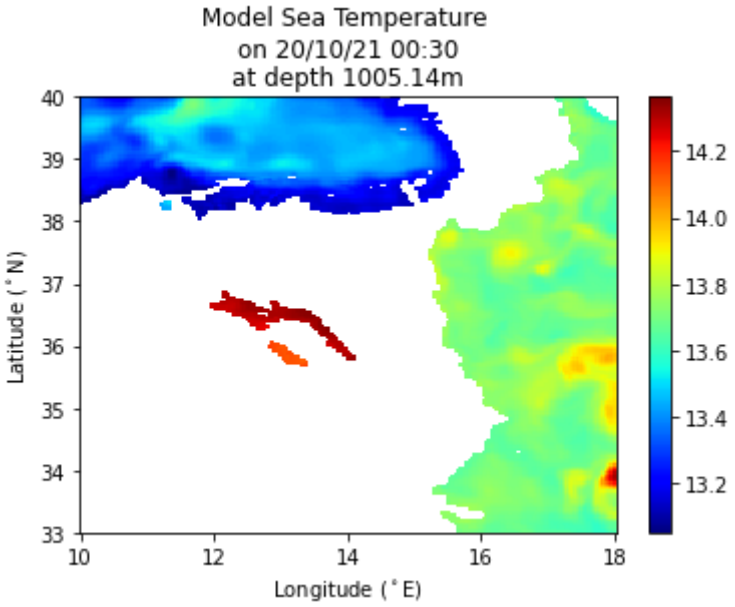


Processing frames across depth

Looping over all depth levels, for the first time frame, and showing the whole domain.

In [17]:

```
nDepthFrames = modelTemperature.shape[1];

for d in range(nDepthFrames):

    t = 0;
    timeMins = float(modelTime[t])
    modelTimeCurr = modelBaseDate + datetime.timedelta(minutes = timeMins)

    modelTemp = modelTemperature[t,d,:,:];

    plt.clf()
    plt.pcolor(modelLon, modelLat, modelTemp, shading='auto');
    plt.ylabel("Latitude ($^\circ$N)");
    plt.xlabel("Longitude ($^\circ$E)");
    plt.title("Model Sea Temperature \non " +  modelTimeCurr.strftime("%d/%m/%y %H:%
    plt.colorbar();
    #plt.clim([18, 26]);
    display.display(plt.gcf())
    display.clear_output(wait=True)
    #time.sleep(0.25)
```

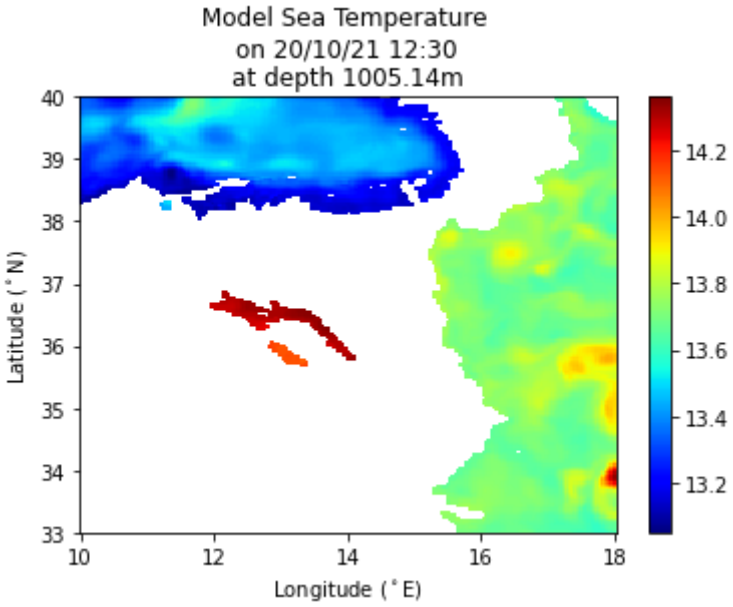


Exercise 3 (replace ?? with the correct terms):

- Define the number of depth frames (nDepthFrames)
- Identify the time frame corresponding to 20/12/21 12:30 (t)
- Loop over all depth levels. Show the whole domain.
- Make sure to display the colorbar.

In [18]:

```
nDepthFrames = modelTemperature.shape[1];

for d in range(nDepthFrames):

    t = 12;
    timeMins = float(modelTime[t])
    modelTimeCurr = modelBaseDate + datetime.timedelta(minutes = timeMins)

    modelTemp = modelTemperature[t,d,:,:];

    plt.clf()
    plt.pcolor(modelLon, modelLat, modelTemp, shading='auto');
    plt.ylabel("Latitude ($^\circ$N)");
    plt.xlabel("Longitude ($^\circ$E)");
    plt.title("Model Sea Temperature \non " +  modelTimeCurr.strftime("%d/%m/%y %H:%
    plt.colorbar();
    display.display(plt.gcf())
    display.clear_output(wait=True)
    #time.sleep(0.25)
```



Extracting time series from model data

Defining required coordinates.

In [19]:
```
reqLon = 16.0;
reqLat = 36.0;
```

Finding closest longitude in model domain.

In [20]:
```
minVal = 9999;
minValPos = 0;
for i in range(modelLon.shape[0]):
    currLon = float (modelLon[i])
    currTest = abs(currLon - reqLon)
    if (currTest < minVal):
        minVal = currTest
        minValPos = i
modelLonIdx = minValPos
```

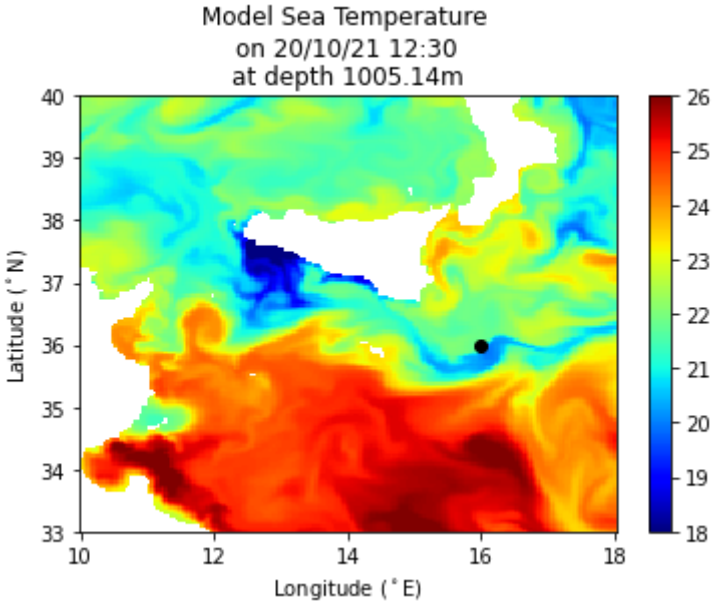
Finding closest latitude in model domain.

In [21]:
```
minVal = 9999;
minValPos = 0;
for i in range(modelLat.shape[0]):
    currLat = float (modelLat[i])
    currTest = abs(currLat - reqLat)
    if (currTest < minVal):
        minVal = currTest
        minValPos = i
modelLatIdx = minValPos
```

Showing Point on map.

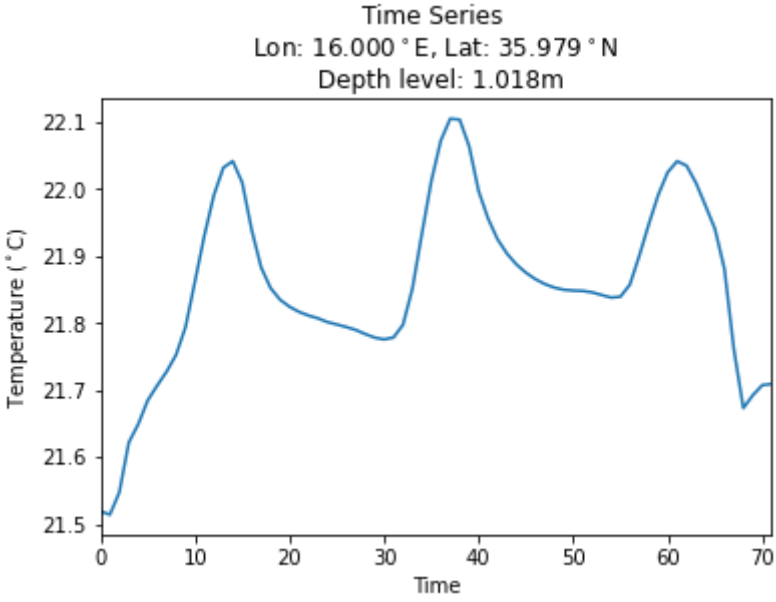
In [22]:
```
modelTemp = modelTemperature[0,0,:,:];
plt.pcolor(modelLon, modelLat, modelTemp, shading='auto');
plt.plot(modelLon[modelLonIdx], modelLat[modelLatIdx], 'ko')
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Model Sea Temperature \non " +  modelTimeCurr.strftime("%d/%m/%y %H:%M")
```

```
plt.colorbar();
plt.clim([18, 26]);
```



Extracting and plotting time series.

In [23]:
```
depthLevel = 0;

modelTimeSeries = modelTemperature[:,depthLevel,modelLatIdx,modelLonIdx];

plt.plot(modelTimeSeries);
plt.ylabel("Temperature ($^\circ$C)")
plt.xlabel("Time")
plt.title("Time Series \nLon: " + "{:.3f}".format(modelLon[modelLonIdx]) + "$^\circ$
plt.autoscale(enable=True, axis='x', tight=True)
```



Exercise 4 (replace ?? with the correct terms):

- Compute the sea temperature predicted by the model at the 36th time frame, for the first depth level (1.018m) at 16.0 degE in longitude and 36.0 degN in latitude;

In [24]:
```
depthLevel = 0;
```

```
timeFrame = 36;

timeMins = float(modelTime[timeFrame])
modelTimeCurr = modelBaseDate + datetime.timedelta(minutes = timeMins)

depthCurr = modelDepth[depthLevel];

modelTempValue = modelTemperature[timeFrame,depthLevel,modelLatIdx,modelLonIdx];

print("Model temperature at " + modelTimeCurr.strftime("%d/%m/%y %H:%M") + " at " +
```

```
Model temperature at 21/10/21 12:30 at 1.018m is 22.072296deg C
```

Quiz Question 2:

What is the sea temperature predicted by the model for 21/10/2021 06:30 at the first depth level (1.018m) at 16.0 degE in longitude and 36.0 degN in latitude:

1. 19.82 degC
2. 20.13 degC
3. 21.78 degC
4. 22.52 degC

Extracting depth profile from model data

Defining required coordinates.

In [25]:

```
reqLon = 14.0;
reqLat = 36.5;
```

Finding closest longitude in model domain.

In [26]:

```
minVal = 9999;
minValPos = 0;
for i in range(modelLon.shape[0]):
    currLon = float (modelLon[i])
    currTest = abs(currLon - reqLon)
    if (currTest < minVal):
        minVal = currTest
        minValPos = i
modelLonIdx = minValPos
```

Finding closest latitude in model domain.

In [27]:

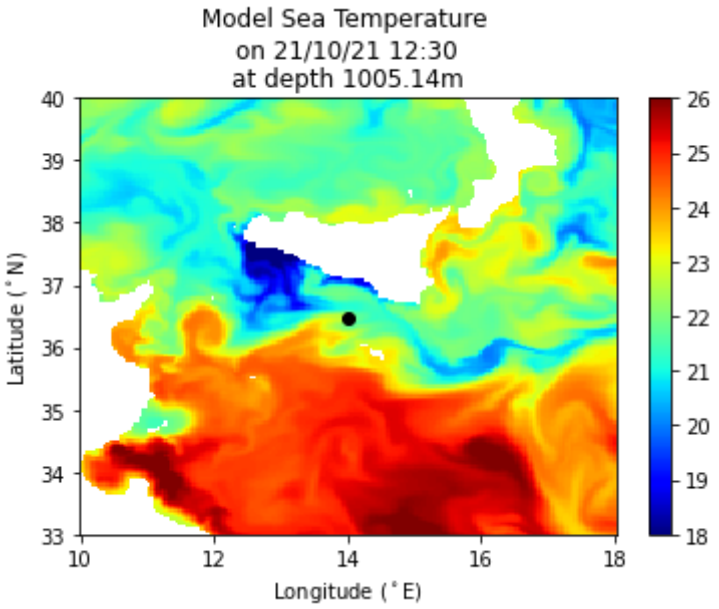
```
minVal = 9999;
minValPos = 0;
for i in range(modelLat.shape[0]):
    currLat = float (modelLat[i])
    currTest = abs(currLat - reqLat)
    if (currTest < minVal):
        minVal = currTest
        minValPos = i
modelLatIdx = minValPos
```

Showing Point on map.

In [28]:

```
modelTemp = modelTemperature[0,0,:,:];
```

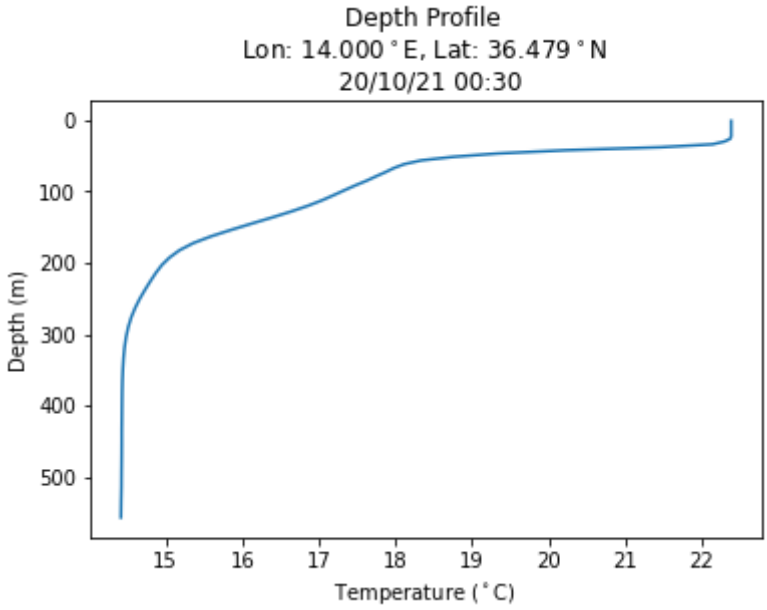
```
plt.pcolor(modelLon, modelLat, modelTemp, shading='auto');
plt.plot(modelLon[modelLonIdx], modelLat[modelLatIdx], 'ko')
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Model Sea Temperature \non " +  modelTimeCurr.strftime("%d/%m/%y %H:%M")
plt.colorbar();
plt.clim([18, 26]);
```



Extracting and plotting depth profile.

In [29]:

```
timeStep = 0;
timeMins = float(modelTime[timeStep]);
modelTimeStep = modelBaseDate + datetime.timedelta(minutes = timeMins);

modelDepthProfile = modelTemperature[timeStep,:,modelLatIdx,modelLonIdx];

plt.plot(modelDepthProfile, modelDepth);
plt.gca().invert_yaxis();
plt.ylabel("Depth (m)")
plt.xlabel("Temperature ($^\circ$C)")
plt.title("Depth Profile \nLon: " + "{:.3f}".format(modelLon[modelLonIdx]) + "$^\cir
```



Processing satellite data

Reading satellite netcdf file.

In [30]:
```
satFilename = 'Data/SST_MED_SST_L4_NRT_OBSERVATIONS_010_004_c_V2_1634969733519.nc'
satDataSet = nc.Dataset(satFilename)
```

Printing metadata of the netcdf file and of other parameters.

In [31]:
```
print(satDataSet)
```

```
<class 'netCDF4._netCDF4.Dataset'>
root group (NETCDF3_CLASSIC data model, file format NETCDF3):
    Conventions: CF-1.4
    title: Mediterranean SST Analysis, L4, 1km daily (SST_MED_SST_L4_NRT_OBSERVATION
S_010_004_c_V2)
    summary: Daily gap-free maps (L4) at 1/100deg. x 1/100deg. horizontal resolution
over the Mediterranean Sea.The data are obtained from infra-red measurements collect
ed by satellite radiometers and statistical interpolation.It is the Copernicus sea s
urface temperature nominal operational product for the Mediterranean Sea in NRT mod
e.
    references: Buongiorno Nardelli, B., C. Tronconi, a. Pisano, and R. Santoleri, 2
013: High and Ultra-High resolution processing of satellite Sea Surface Temperature
data over Southern European Seas in the framework of MyOcean project. /Remote Sens.
Environ./, *129*, 1-16, doi:10.1016/j.rse.2012.10.012. AND Buongiorno Nardelli, B.,
a. Pisano, C. Tronconi, and R. Santoleri, 2015: Evaluation of different covariance m
odels for the operational interpolation of high resolution satellite Sea Surface Tem
perature data over the Mediterranean Sea. /Remote Sens. Environ./, doi:10.1016/j.rs
e.2015.04.025.
    institution: GOS
    history: GOS-MYO processor V3: new version
    comment: WARNING: some applications are unable to properly handle byte values. I
f Values >127 are encounterd, please subtract 256
    license: free registration at Copernicus Marine Service (http://marine.copernicu
s.eu/web/56-user-registration-form.php)
    id: OISST_UHR_NRT-GOS-L4-MED-v1.0
    DSD_entry_id: -GOS-L4UHRfnd-MED
    naming_authority: org.ghrsst
    product_version: 1.0
    uuid:
    gds_version_id: v2.0.5
    netcdf_version_id: 3.6.0-p1 of Oct 16 2005 13:23:24
    date_created: 20211022T110226Z
    file_quality_level: 0
    spatial_resolution: 0.01 degree
    start_time: 20211021T190000Z
    time_coverage_start: 20211021T190000Z
    stop_time: 20211022T060000Z
    time_coverage_end: 20211022T060000Z
    software_version: Copernicus Marine Service UHR L4 Processor V.1
    westernmost_longitude: -18.120832
    easternmost_longitude: 36.245834
    southernmost_latitude: 30.254168
    northernmost_latitude: 45.995834
    Scaling_Equation: (scale_factor*data) + add_offset
    source: EUR-L2P-ATS_NR_2P, UPA-L2P-ATS_NR_2P, JPL-L2P-MODIS_A, JPL-L2P-MODIS_T,
EUR-L2P-NAR17_SST, EUR-L2P-NAR18_SST, EUR-L2P-SEVIRI_SST, EUR-L2P-AVHRR_METOP_A, EUR
-L2P-AVHRR_METOP_B, NAVO-L2P-VIIRS_NPP, EUR-L2P-SEVIRI_SST_Meteosat11, MAR-L2P-SLSTR
_ S3A,MAR-L2P-SLSTR_S3B
    platform: Envisat, NOAA-18, MetOpA, Aqua, Terra, MSG1, MetOpB, NPP, Meteosat11,
Sentinel 3A, Sentinel 3B
    sensor: AATSR, SEVIRI, AVHRR, MODIS, VIIRS, SLSTR
```

```
    Metadata_Conventions: Unidata Dataset Discovery v1.0
    metadata_link: Link to collection metadata record at archive
    keywords: Oceans > Ocean Temperature > Sea Surface Temperature
    keywords_vocabulary: NASA Global Change Master Directory (GCMD) Science Keywords
    standard_name_vocabulary: NetCDF Climate and Forecast (CF) Metadata Convention
    geospatial_lat_units: degrees_north
    geospatial_lat_resolution: 0.01
    geospatial_lon_units: degrees_east
    geospatial_lon_resolution: 0.01
    acknowledgment: Please acknowledge the use of these data with the following stat
ement: Generated/provided by Copernicus Marine Service and CNR - ISMAR ROME. We woul
d also appreciate being informed of any publications.
    creator_name: ISMAR - Institute of Marine Sciences (CNR - Rome)
    creator_email: gsdk@isac.cnr.it
    creator_url: http://gosweb.artov.isac.cnr.it/
    project: Copernicus Marine Service
    publisher_name: CNR ISMAR GOS - WITS Copernicus Marine Service
    publisher_url: http://marine.copernicus.eu/
    publisher_email: servicedesk.cmems@mercator-ocean.eu, gsdk@isac.cnr.it
    processing_level: L4
    cdm_data_type: grid
    History: Translated to CF-1.0 Conventions by Netcdf-Java CDM (CFGridWriter2)
Original Dataset = SST_MED_SST_L4_NRT_OBSERVATIONS_010_004_c_V2; Translation Date =
2021-10-23T06:15:33.578Z
    geospatial_lat_min: 32.99583435058594
    geospatial_lat_max: 39.99583435058594
    geospatial_lon_min: 9.99583625793457
    geospatial_lon_max: 17.99583625793457
    dimensions(sizes): time(3), lat(841), lon(961)
    variables(dimensions): int16 analysis_error(time, lat, lon), int32 time(time), f
loat32 lat(lat), float32 lon(lon), int16 analysed_sst(time, lat, lon)
    groups:
```

In [33]:

```
print (satDataSet['analysed_sst'])
```

```
<class 'netCDF4._netCDF4.Variable'>
int16 analysed_sst(time, lat, lon)
    long_name: analysed sea surface temperature
    standard_name: sea_surface_temperature
    type: foundation
    units: kelvin
    _FillValue: -32768
    add_offset: 273.15
    scale_factor: 0.01
    valid_min: -300
    valid_max: 4500
    source: EUR-L2P-ATS_NR_2P,UPA-L2P-ATS_NR_2P,JPL-L2P-MODIS_A,JPL-L2P-MODIS_T,EUR-
L2P-NAR17_SST,EUR-L2P-NAR18_SST,EUR-L2P-SEVIRI_SST,EUR-L2P-AVHRR_METOP_A,EUR-L2P-AVH
RR_METOP_B,NAVO-L2P-VIIRS_NPP,EUR-L2P-SEVIRI_SST_Meteosat11,MAR-L2P-SLSTR_S3A,MAR-L2
P-SLSTR_S3B
    comment: Optimal interpolation (OI) at Ultrahigh resolution of sst measurements
from infrared sensors
    coordinates: time lat lon
unlimited dimensions:
current shape = (3, 841, 961)
filling on
```

In [34]:

```
print (satDataSet['time'])
```

```
<class 'netCDF4._netCDF4.Variable'>
int32 time(time)
    long_name: reference time of sst field
    standard_name: time
```

```
    axis: T
    calendar: Gregorian
    units: seconds since 1981-01-01 00:00:00
    comment: Nominal time of Level 4 analysis
    _CoordinateAxisType: Time
unlimited dimensions:
current shape = (3,)
filling on, default _FillValue of -2147483647 used
```

Reading the variables from the satellite nc file.

In [35]:
```
satLon = satDataSet['lon']
satLat = satDataSet['lat']
satTemperature = satDataSet['analysed_sst']
satTime = satDataSet['time']
```

Extracting temperature values across all longitudes and all latitudes, for the first time frame (out of 72), and converting values from Kelvin to Celcius.

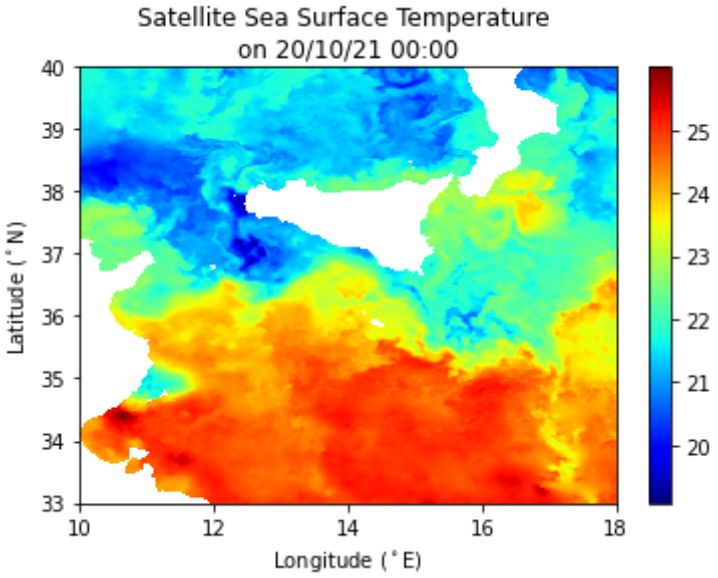
In [36]:
```
satTemp1 = satTemperature[0,:,:];
satTemp1 -= 273.15;
```

Calculating the time dorresponding to this frame.

In [37]:
```
satBaseDate =  datetime.datetime(1981, 1, 1, 0, 0, 0);

timeSecs = float(satTime[0])
satTimeDT1 = satBaseDate + datetime.timedelta(seconds = timeSecs)
```

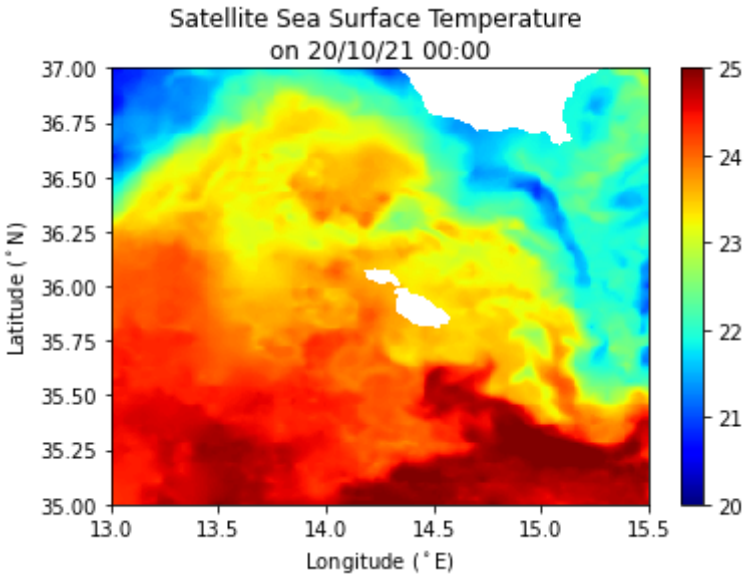
Plotting the extracted satellite data (satTemp1).

In [38]:
```
plt.pcolor(satLon, satLat, satTemp1, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Satellite Sea Surface Temperature \non " +  satTimeDT1.strftime("%d/%m/%y
plt.colorbar();
```



Plotting the extracted temperature data (satTemp1), zooming on Malta, and adjusting the colorbar limits accordingly.

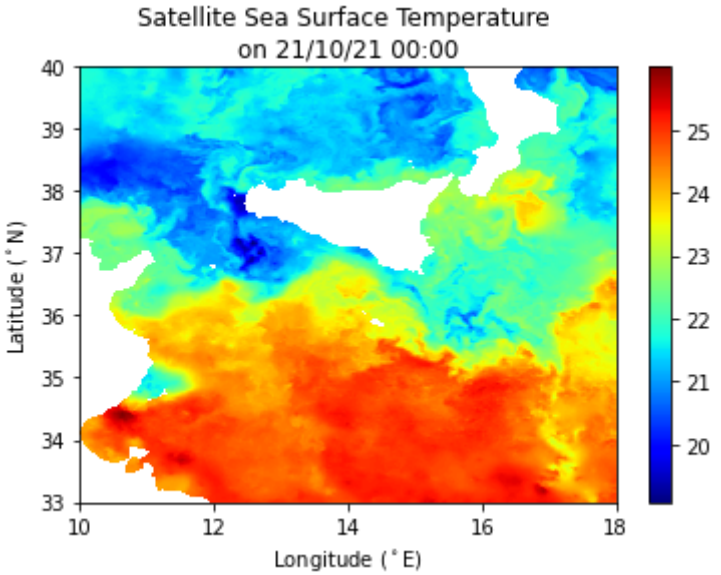
In [39]:
```
plt.pcolor(satLon, satLat, satTemp1, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Satellite Sea Surface Temperature \non " +  satTimeDT1.strftime("%d/%m/%y
plt.axis([13, 15.5, 35, 37]);
plt.colorbar();
plt.clim([20, 25]);
```



Exercise 5 (replace ?? with the correct terms):

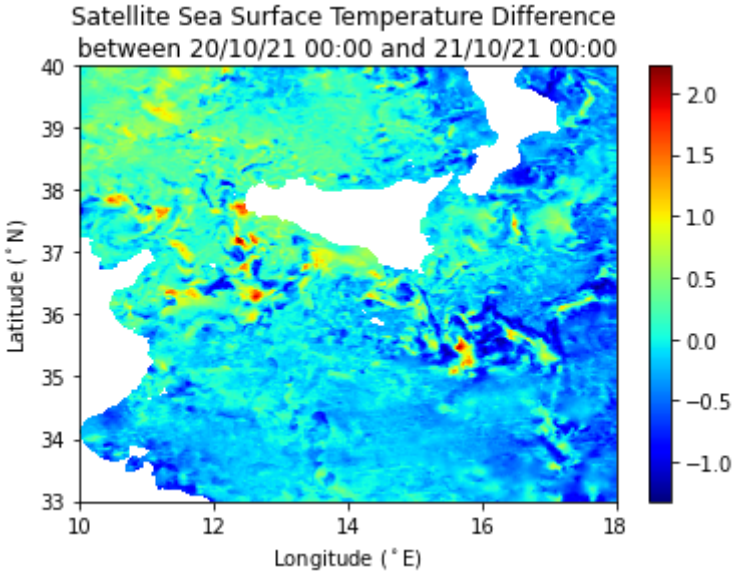
- Extract satellite temperature values across all longitudes and all latitudes, for the second time frame
- Convert temperature values from Kelvin to Celcius
- Calculate the time corresponding to this frame
- Plot the extracted temperature data (satTemp2)

In [40]:
```
satTemp2 = satTemperature[1,:,:];
satTemp2 -= 273.15;

timeSecs = float(satTime[1])
satTimeDT2 = satBaseDate + datetime.timedelta(seconds = timeSecs)

plt.pcolor(satLon, satLat, satTemp1, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Satellite Sea Surface Temperature \non " +  satTimeDT2.strftime("%d/%m/%y
plt.colorbar();
```



Compute and plot difference in temperature between the two days.

In [41]:
```
satDiff = satTemp2 - satTemp1;

plt.pcolor(satLon, satLat, satDiff, shading='auto');
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Satellite Sea Surface Temperature Difference \nbetween " +  satTimeDT1.st
plt.colorbar();
plt.set_cmap('jet')
```



Quiz Question 3:

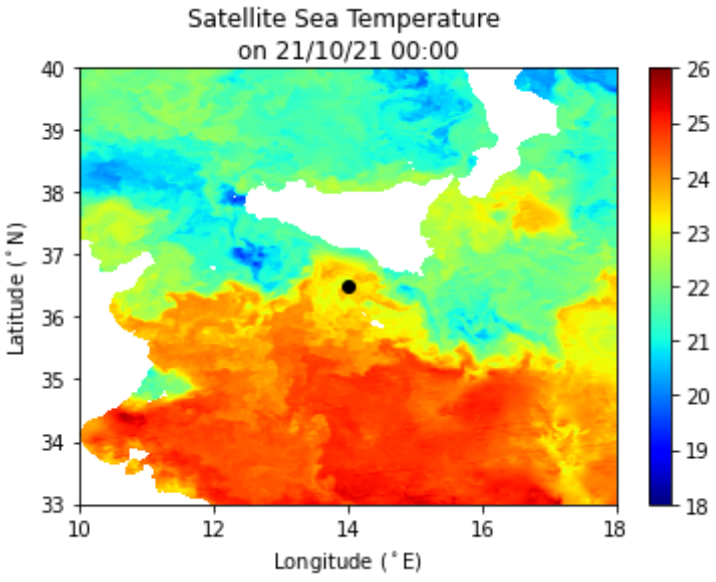
What is the sea temperature recorded by the satellite for 21/10/2021 00:00 at 14.0 degE in longitude and 36.5 degN in latitude:

1. 18.43 degC
2. 19.78 degC
3. 21.89 degC
4. 22.64 degC

In [42]:
```

```

```
reqLon = 14.0;
reqLat = 36.5;

minVal = 9999;
minValPos = 0;
for i in range(satLon.shape[0]):
    currLon = float (satLon[i])
    currTest = abs(currLon - reqLon)
    if (currTest < minVal):
        minVal = currTest
        minValPos = i
satLonIdx = minValPos

minVal = 9999;
minValPos = 0;
for i in range(satLat.shape[0]):
    currLat = float (satLat[i])
    currTest = abs(currLat - reqLat)
    if (currTest < minVal):
        minVal = currTest
        minValPos = i
satLatIdx = minValPos

timeSecs = float(satTime[1])
satTimeDT = satBaseDate + datetime.timedelta(seconds = timeSecs)

satTemp = satTemperature[1,:,:];
satTemp -= 273.15;

plt.pcolor(satLon, satLat, satTemp, shading='auto');
plt.plot(satLon[satLonIdx], satLat[satLatIdx], 'ko')
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Satellite Sea Temperature \non " +  satTimeDT.strftime("%d/%m/%y %H:%M"))
plt.colorbar();
plt.clim([18, 26]);

satTempValue = satTemp[satLonIdx, satLatIdx];

print("Satellite temperature at " + satTimeDT.strftime("%d/%m/%y %H:%M") + " is " +
```

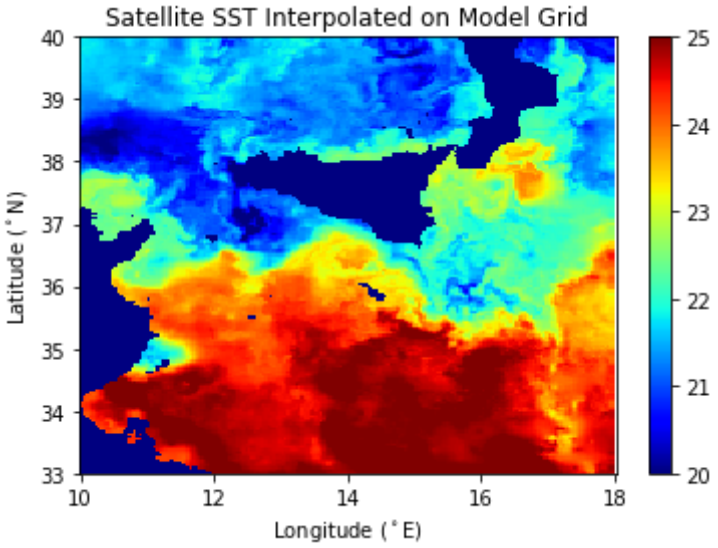
```
Satellite temperature at 21/10/21 00:00 is 22.639978deg C
```



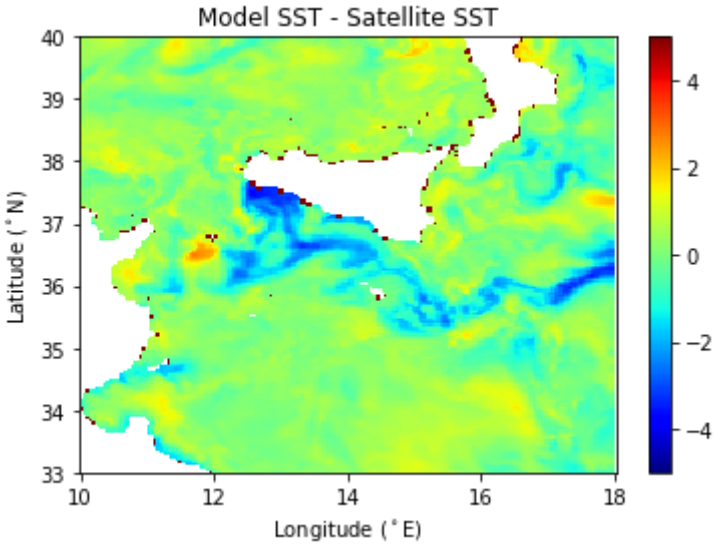
Comparing Satellite and Model Data

In [43]:
```
modelTemp = modelTemperature[0,0,:,:];

satTemp = satTemperature[0,:,:];
satTemp -= 273.15;

xi, yi = np.meshgrid(satLon, satLat)
Xi, Yi = np.meshgrid(modelLon, modelLat)

satTempInterp = interp.griddata((xi.flatten(), yi.flatten()), satTemp.flatten(), (Xi
satTempInterp = np.reshape(satTempInterp, Xi.shape)
```

In [44]:
```
plt.pcolor(modelLon, modelLat, satTempInterp, shading='auto');
plt.clim([20, 25]);
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Satellite SST Interpolated on Model Grid");
plt.colorbar();
```

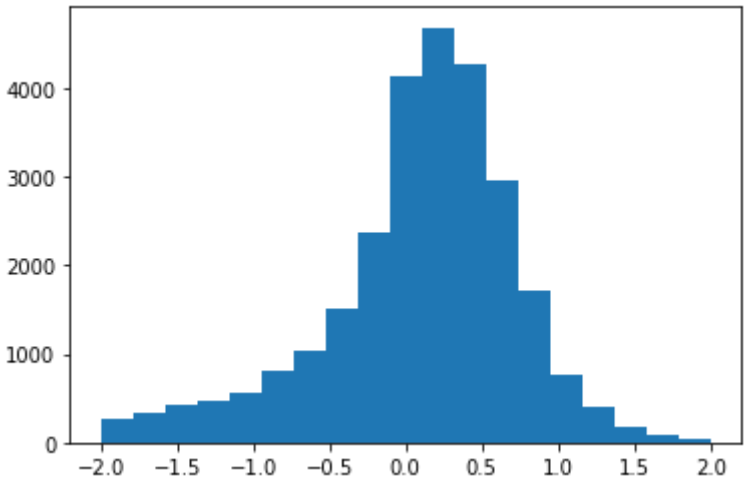


In [45]:
```
tempDiff = modelTemp - satTempInterp
plt.pcolor(modelLon, modelLat, tempDiff, shading='auto');
plt.clim([-5, 5]);
plt.ylabel("Latitude ($^\circ$N)");
plt.xlabel("Longitude ($^\circ$E)");
plt.title("Model SST - Satellite SST");
plt.colorbar();
```



In [46]:
```
x, y, z = plt.hist(tempDiff.flatten(), np.linspace(-2,2,20));
elem = np.argmax(x)
y[elem]
```

Out[46]:
```
0.10526315789473673
```



Quiz Question 4:

What is the most common temperature difference between the model and satellite datasets for 20/10/2021 00:00:

1. 0.07846 degC
2. 0.10526 degC
3. 0.15632 degC
4. 0.18370 degC

In [ ]:
```
```