

```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using CRA.clima.gsrad;
using CRA.clima.gsrad.interfaces;

namespace CRA.clima.webservices.GSRad
{
    /// <summary>
    /// Web service made on the component CRA.clima.GSRad http://www.isci.it/tools
    /// </summary>
    [WebService(Namespace="http://www.sipeaa.it/webservices/gsrad")]
    public class CRA_clima_GSRad : System.Web.Services.WebService
    {
        public CRA_clima_GSRad()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }

        #region Component Designer generated code

        //Required by the Web Services Designer
        private IContainer components = null;

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {

```

```

        components.Dispose();
    }
    base.Dispose(disposing);
}

#endregion

#region Global solar radiation

[WebMethod]
public double GlobalSolarRadiationAngstromPrescott(double slopeAspectFactor,
    double angstromPrescott_minimumTransmissivity, double angstromPrescott_slopeTrasmissivity,
    double sunshineDuration, double clearSkyTransmissivity,
    double dayLenght, double extraterrestrialRadiation)
{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrad.GSRad();
    //create instance of strategy for the GSRad context constructor
    IGlobalSolarRadiation m = new GSRangstromPrescott();
    //select strategy
    rd.ModelGSRad = m;
    //assign parameter of relevant strategy
    GSRangstromPrescott.angstromPrescott_minimumTransmissivity =
        angstromPrescott_minimumTransmissivity;
    GSRangstromPrescott.angstromPrescott_slopeTrasmissivity =
        angstromPrescott_slopeTrasmissivity;

    //assign parameters
    rd.sunshineDuration = sunshineDuration;
    rd.slopeAspectFactor = slopeAspectFactor;
    rd.clearSkyTransmissivity = clearSkyTransmissivity;
    rd.extraterrestrialRadiation = extraterrestrialRadiation;
    rd.dayLenght = dayLenght;
    //call for global solar radiation
    gs.GroundRadiation(ref rd);
    return Math.Round(rd.globalSolarRadiation,2);
}

[WebMethod]
public double GlobalSolarRadiationBristowCampbell(double slopeAspectFactor,
    double bristowCampbell_b, double airTemperatureDailyRange,
    double airTemperatureMonthlyRange, double clearSkyTransmissivity,
    double extraterrestrialRadiation)

```

```

{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrاد.GSRad();
    //create instance of strategy for the GSRad context constructor
    IGlobalSolarRadiation m = new GSRBristowCampbell();
    //select strategy
    rd.ModelGSRad = m;
    //assign parameter of relevant strategy
    GSRBristowCampbell.bristowCampbell_b = bristowCampbell_b;
    //assign parameters
    rd.airTemperatureDailyRange = airTemperatureDailyRange;
    rd.airTemperatureMonthlyRange = airTemperatureMonthlyRange;
    rd.extraterrestrialRadiation = extraterrestrialRadiation;
    rd.slopeAspectFactor = slopeAspectFactor;
    rd.clearSkyTransmissivity = clearSkyTransmissivity;
    //call for global solar radiation
    gs.GroundRadiation(ref rd);
    return Math.Round(rd.globalSolarRadiation,2);
}

```

[WebMethod]

```

public double GlobalSolarRadiationCampbellDonatelli(double slopeAspectFactor,
double campbellDonatelli_b, double campbellDonatelli_Tnc, double airTemperatureDailyRange,
double airTemperatureDailyAverage, double minAirTemperature, double clearSkyTransmissivity,
double extraterrestrialRadiation)

```

```

{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrاد.GSRad();
    //create instance of strategy for the GSRad context constructor
    IGlobalSolarRadiation m = new GSRCampbellDonatelli();
    //select strategy
    rd.ModelGSRad = m;
    //assign parameter of relevant strategy
    GSRCampbellDonatelli.campbellDonatelli_b = campbellDonatelli_b;
    GSRCampbellDonatelli.campbellDonatelli_Tnc = campbellDonatelli_Tnc;
    //assign parameters
    rd.airTemperatureDailyRange = airTemperatureDailyRange;
    rd.airTemperatureDailyAverage = airTemperatureDailyAverage;
    rd.minAirTemperature = minAirTemperature;
    rd.extraterrestrialRadiation = extraterrestrialRadiation;
}

```

```

        rd.slopeAspectFactor = slopeAspectFactor;
        rd.clearSkyTransmissivity = clearSkyTransmissivity;
        //call for global solar radiation
        gs.GroundRadiation(ref rd);
        return Math.Round(rd.globalSolarRadiation,2);
    }

    [WebMethod]
    public double GlobalSolarRadiationDonatelliBellocchi(double slopeAspectFactor,
        double donatelliBellocchi_b, double donatelliBellocchi_c2,
        double donatelliBellocchi_c1, double airTemperatureDailyRange,
        double airTemperatureWeeklyRange, double clearSkyTransmissivity,
        double extraterrestrialRadiation)
    {
        //instance of data-type
        RadData rd = new RadData();
        //instance of the GSRad class containing public methods
        IGSRad gs = new CRA.clima.gsrad.GSRad();
        //create instance of strategy for the GSRad context constructor
        IGlobalSolarRadiation m = new GSRDonatelliBellocchi();
        //select strategy
        rd.ModelGSRad = m;
        //assign parameter of relevant strategy
        GSRDonatelliBellocchi.donatelliBellocchi_b = donatelliBellocchi_b;
        GSRDonatelliBellocchi.donatelliBellocchi_c1 = donatelliBellocchi_c1;
        GSRDonatelliBellocchi.donatelliBellocchi_c2 = donatelliBellocchi_c2;
        //assign inputs
        rd.extraterrestrialRadiation = extraterrestrialRadiation;
        rd.airTemperatureDailyRange = airTemperatureDailyRange;
        rd.airTemperatureWeeklyRange = airTemperatureWeeklyRange;
        rd.slopeAspectFactor = slopeAspectFactor;
        rd.clearSkyTransmissivity = clearSkyTransmissivity;
        //call for global solar radiation
        gs.GroundRadiation(ref rd);
        return Math.Round(rd.globalSolarRadiation,2);
    }

    [WebMethod]
    public double GlobalSolarRadiationJohnsonWoodward(double slopeAspectFactor,
        double johnsonWoodward_cloudBlueFactor,
        double[] solarElevation, double angleSlope,
        double clearSkyTransmissivity,
        double extraterrestrialRadiation,
        double sunshineDuration)

```

```

{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrاد.GSRad();
    //create instance of strategy for the GSRad context constructor
    IGlobalSolarRadiation m = new GSRJohnsonWoodward();
    //select strategy
    rd.ModelGSRad = m;
    //assign parameter of relevant strategy
    GSRJohnsonWoodward.johnsonWoodward_cloudBlueFactor = johnsonWoodward_cloudBlueFactor;
    //assign inputs
    rd.extraterrestrialRadiation = extraterrestrialRadiation;
    rd.angleSlope = angleSlope;
    rd.solarElevation = solarElevation;
    rd.slopeAspectFactor = slopeAspectFactor;
    rd.clearSkyTransmissivity = clearSkyTransmissivity;
    rd.sunshineDuration = sunshineDuration;
    //call for global solar radiation
    gs.GroundRadiation(ref rd);
    return Math.Round(rd.globalSolarRadiation,2);
}

[WebMethod]
public double GlobalSolarRadiationSupitVanKappel(double slopeAspectFactor,
    double supitVanKappel_cloudCoverCoefficient, double supitVanKappel_adjustmentFactor,
    double supitVanKappel_dailyCloudCover, double supitVanKappel_temperatureRangeCoefficient,
    double airTemperatureDailyRange, double clearSkyTransmissivity,
    double extraterrestrialRadiation)
{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrاد.GSRad();
    //create instance of strategy for the GSRad context constructor
    IGlobalSolarRadiation m = new GSRSupitVanKappel();
    //select strategy
    rd.ModelGSRad = m;
    //assign parameter of relevant strategy
    GSRSupitVanKappel.supitVanKappel_adjustmentFactor = supitVanKappel_adjustmentFactor;
    GSRSupitVanKappel.supitVanKappel_cloudCoverCoefficient = supitVanKappel_cloudCoverCoefficient;
    GSRSupitVanKappel.supitVanKappel_dailyCloudCover = supitVanKappel_dailyCloudCover;
    GSRSupitVanKappel.supitVanKappel_temperatureRangeCoefficient =
        supitVanKappel_temperatureRangeCoefficient;
}

```

```

        //assign inputs
        rd.airTemperatureDailyRange = airTemperatureDailyRange;
        rd.slopeAspectFactor = slopeAspectFactor;
        rd.clearSkyTransmissivity = clearSkyTransmissivity;
        rd.extraterrestrialRadiation = extraterrestrialRadiation;
        //call for global solar radiation
        gs.GroundRadiation(ref rd);
        return Math.Round(rd.globalSolarRadiation,2);
    }

    [WebMethod]
    public double GlobalSolarRadiationWinslowEtAl(double airTemperatureYearlyAverage,
        double airTemperatureYearlyRange, double clearSkyTransmissivity,
        double dayLenght, double extraterrestrialRadiation,
        double maxAirTemperature, double minAirTemperature, double slopeAspectFactor)
    {
        //instance of data-type
        RadData rd = new RadData();
        //instance of the GSRad class containing public methods
        IGSRad gs = new CRA.clima.gsrad.GSRad();
        //create instance of strategy for the GSRad context constructor
        IGlobalSolarRadiation m = new GSRWinslowEtAl();
        //select strategy
        rd.ModelGSRad = m;
        //assign parameter of relevant strategy
        // -- none for this strategy --
        //assign inputs
        rd.airTemperatureYearlyAverage = airTemperatureYearlyAverage;
        rd.airTemperatureYearlyRange = airTemperatureYearlyRange;
        rd.clearSkyTransmissivity = clearSkyTransmissivity;
        rd.extraterrestrialRadiation = extraterrestrialRadiation;
        rd.maxAirTemperature = maxAirTemperature;
        rd.minAirTemperature = minAirTemperature;
        rd.slopeAspectFactor = slopeAspectFactor;
        //call for global solar radiation
        gs.GroundRadiation(ref rd);
        return Math.Round(rd.globalSolarRadiation,2);
    }

    [WebMethod]
    public double GlobalSolarRadiationRichardson(double globalSolarRadiationResidual,
        double numberOfDaysInMonth, double numberOfWetDaysInMonth, double radiationMonthlyAverage,
        double radiationMonthlyStandardDeviation, double scalingFactorWetDry)
    {

```

```

//instance of data-type
RadData rd = new RadData();
//instance of the GSRad class containing public methods
IGSRad gs = new CRA.clima.gsrاد.GSRad();
//create instance of strategy for the GSRad context constructor
IGlobalSolarRadiation m = new GSRRichardson();
//select strategy
rd.ModelGSRad = m;
//assign parameter of relevant strategy
CRA.clima.gsrاد.GSRRichardson.richardson_numberOfDaysInMonth = numberOfDaysInMonth;
CRA.clima.gsrاد.GSRRichardson.richardson_numberOfWetDaysInMonth = numberOfWetDaysInMonth;
CRA.clima.gsrاد.GSRRichardson.richardson_radiationMonthlyAverage = radiationMonthlyAverage;
CRA.clima.gsrاد.GSRRichardson.richardson_radiationMonthlyStandardDeviation =
radiationMonthlyStandardDeviation;
CRA.clima.gsrاد.GSRRichardson.richardson_scalingFactorWetDry = scalingFactorWetDry;
//assign parameters
rd.globalSolarRadiationResidual = globalSolarRadiationResidual;
//call for extraterrestrial radiation
gs.GroundRadiation(ref rd);
return Math.Round(rd.globalSolarRadiation,2);
}

```

[WebMethod]

```

public double GlobalSolarRadiationGarciaHogenboom(double globalSolarRadiationResidual,
double garciaHogenboom_adjustmentFactorGBARDry,
double garciaHogenboom_adjustmentFactorGBARWet,
double garciaHogenboom_adjustmentFactorGNoise, bool garciaHogenboom_dayStatusWet,
double garciaHogenboom_gsrYearlyAmplitudeDry, double garciaHogenboom_gsrYearlyAmplitudeWet,
double garciaHogenboom_gsrYearlyMeanDry, double garciaHogenboom_gsrYearlyMeanWet)

```

{

```

//instance of data-type
RadData rd = new RadData();
//instance of the GSRad class containing public methods
IGSRad gs = new CRA.clima.gsrاد.GSRad();
//create instance of strategy for the GSRad context constructor
IGlobalSolarRadiation m = new GSRGarciaHogenboom();
//select strategy
rd.ModelGSRad = m;
//assign parameter of relevant strategy
CRA.clima.gsrاد.GSRGarciaHogenboom.garciaHogenboom_adjustmentFactorGBARDry =
garciaHogenboom_adjustmentFactorGBARDry;
CRA.clima.gsrاد.GSRGarciaHogenboom.garciaHogenboom_adjustmentFactorGBARWet =
garciaHogenboom_adjustmentFactorGBARWet;
CRA.clima.gsrاد.GSRGarciaHogenboom.garciaHogenboom_adjustmentFactorGNoise =

```

```

        garciaHogenboom_adjustmentFactorGNoise;
CRA.clima.gsrad.GSRGarciaHogenboom.garciaHogenboom_dayStatusWet =
        garciaHogenboom_dayStatusWet;
CRA.clima.gsrad.GSRGarciaHogenboom.garciaHogenboom_gsrYearlyAmplitudeDry =
        garciaHogenboom_gsrYearlyAmplitudeDry;
CRA.clima.gsrad.GSRGarciaHogenboom.garciaHogenboom_gsrYearlyAmplitudeWet =
        garciaHogenboom_gsrYearlyAmplitudeWet;
CRA.clima.gsrad.GSRGarciaHogenboom.garciaHogenboom_gsrYearlyMeanDry =
        garciaHogenboom_gsrYearlyMeanDry;
CRA.clima.gsrad.GSRGarciaHogenboom.garciaHogenboom_gsrYearlyMeanWet =
        garciaHogenboom_gsrYearlyMeanWet;

//assign parameters
rd.globalSolarRadiationResidual = globalSolarRadiationResidual;
//call for extraterrestrial radiation
gs.GroundRadiation(ref rd);
return Math.Round(rd.globalSolarRadiation,2);
}
#endregion

#region Global solar radiation residual
[WebMethod]
public double ResidualGlobalSolarRadiation(double[] richardson_A, double[] richardson_B,
double richardson_airTemperatureMaxDayBeforeResidual,
double richardson_airTemperatureMinDayBeforeResidual,
double richardson_radiationDayBeforeResidual,
bool richardson_matrixDataFromSeparatedWetDryDays,
bool richardson_wetDay)
{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrad.GSRad();
    //assign model parameter
    // A matrix
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[0,0] = richardson_A[0];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[0,1] = richardson_A[1];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[0,2] = richardson_A[2];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[1,0] = richardson_A[3];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[1,1] = richardson_A[4];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[1,2] = richardson_A[5];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[2,0] = richardson_A[6];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[2,1] = richardson_A[7];
    CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_A[2,2] = richardson_A[8];
    // B matrix

```

```

CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[0,0] = richardson_B[0];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[0,1] = richardson_B[1];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[0,2] = richardson_B[2];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[1,0] = richardson_B[3];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[1,1] = richardson_B[4];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[1,2] = richardson_B[5];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[2,0] = richardson_B[6];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[2,1] = richardson_B[7];
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_B[2,2] = richardson_B[8];
//Global solar radiation residual parameters
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_airTemperatureMaxDayBeforeResidual =
    richardson_airTemperatureMaxDayBeforeResidual;
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_airTemperatureMinDayBeforeResidual =
    richardson_airTemperatureMinDayBeforeResidual;
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_radiationDayBeforeResidual =
    richardson_radiationDayBeforeResidual;
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_matrixDataFromSeparatedWetDryDays =
    richardson_matrixDataFromSeparatedWetDryDays;
CRA.clima.gsrad.GlobalSolarRadiationResidual.richardson_wetDay = richardson_wetDay;

//call for globl solar radiation residual
gs.GSRadiationResidual(ref rd);
return Math.Round(rd.globalSolarRadiationResidual,2);
}
#endregion

#region ExtraterrestrialRadiation
[WebMethod]
public double ExtraterrestrialRadiationDaily(int currentDay, double latitude)
{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrad.GSRad();
    //assign parameters
    rd.latitude = latitude;
    rd.currentDay = currentDay;
    //call for extraterrestrial radiation
    gs.ExtraterrestrialRadiation(ref rd);
    return Math.Round(rd.extraterrestrialRadiation,2);
}

[WebMethod]
public double[] ExtraterrestrialRadiationHourly(int currentDay, double latitude)

```

```

{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrاد.GSRad();
    //assign parameters
    rd.latitude = latitude;
    rd.currentDay = currentDay;
    //call for extraterrestrial radiation
    gs.ExtraterrestrialRadiation(ref rd);
    return rd.extraterrestrialRadiationHourly;
}
#endregion

#region Slope, Aspect and SlopeAspect factor
[WebMethod]
public double[] SlopeAspect(double cellSize, double[] elevation)
{
    //return matrix
    double[] slopeAspect = new double[2];
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrاد.GSRad();
    //assign parameters
    rd.cellSize = cellSize;
    rd.elevation[0,0] = elevation[0];
    rd.elevation[0,1] = elevation[1];
    rd.elevation[0,2] = elevation[2];
    rd.elevation[1,0] = elevation[3];
    rd.elevation[1,1] = elevation[4];
    rd.elevation[1,2] = elevation[5];
    rd.elevation[2,0] = elevation[6];
    rd.elevation[2,1] = elevation[7];
    rd.elevation[2,2] = elevation[8];
    //call for slope and aspect
    gs.SlopeAspect(ref rd);
    slopeAspect[0] = rd.angleSlope;
    slopeAspect[1] = rd.angleAspect;
    return slopeAspect;
}

[WebMethod]
public double SlopeAspectFactor(double latitude, double angleAspect, double angleSlope,

```

```

    double solarDeclination, double hourSunrise, double hourSunset,
    double[] hourAngle, double[] solarElevation)
{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrad.GSRad();
    //assign parameters
    rd.latitude = latitude;
    rd.angleAspect = angleAspect;
    rd.angleSlope = angleSlope;
    rd.solarDeclination = solarDeclination;
    rd.hourSunrise = hourSunrise;
    rd.hourSunset = hourSunset;
    rd.hourAngle = hourAngle;
    rd.solarElevation = solarElevation;
    //call for slopeAspectFactor
    gs.SlopeAspectFactor(ref rd);
    return rd.slopeAspectFactor;
}

[WebMethod]
public double[] SlopeAspectFactorHourly(double latitude, double angleAspect, double angleSlope,
    double solarDeclination, double hourSunrise, double hourSunset,
    double[] hourAngle, double[] solarElevation)
{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrad.GSRad();
    //assign parameters
    rd.latitude = latitude;
    rd.angleAspect = angleAspect;
    rd.angleSlope = angleSlope;
    rd.solarDeclination = solarDeclination;
    rd.hourSunrise = hourSunrise;
    rd.hourSunset = hourSunset;
    rd.hourAngle = hourAngle;
    rd.solarElevation = solarElevation;
    //call for slopeAspectFactor
    gs.SlopeAspectFactor(ref rd);
    return rd.slopeAspectFactorHourly;
}
#endregion

```

```

#region PAR and PAR components
[WebMethod]
public double PARDaily(double globalSolarRadiation, double radiationBeam,
    double radiationDiffuseSky,
    double ross_a, double ross_b)
{
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrad.GSRad();
    //assign parameters
    CRA.clima.gsrad.PhotosyntheticallyActiveRadiation.ross_a = ross_a;
    CRA.clima.gsrad.PhotosyntheticallyActiveRadiation.ross_b = ross_b;
    //assign inputs
    rd.globalSolarRadiation = globalSolarRadiation ;
    rd.radiationBeam = radiationBeam;
    rd.radiationDiffuseSky = radiationDiffuseSky;
    //call for extraterrestrial radiation
    gs.PhotosyntheticallyActiveRadiation(ref rd);
    return Math.Round(rd.photosyntheticallyActiveRadiation, 3);
}

[WebMethod]
public double[] PARHourly(double globalSolarRadiation, double[] solarElevation,
    double radiationDiffuseSky,
    double supitVanDerGroot_fractionGSRad)
{
    //return matrix
    double[] PARHourly = new double[24];
    //instance of data-type
    RadData rd = new RadData();
    //instance of the GSRad class containing public methods
    IGSRad gs = new CRA.clima.gsrad.GSRad();
    //assign parameters
    CRA.clima.gsrad.PARHourlyBeamDiffuse.supitVanDerGroot_fractionGSRad =
        supitVanDerGroot_fractionGSRad;

    //assign inputs
    rd.globalSolarRadiation = globalSolarRadiation;
    rd.solarElevation = solarElevation;
    rd.radiationDiffuseSky = radiationDiffuseSky;
    //call for extraterrestrial radiation
    gs.PhotosyntheticallyActiveRadiation(ref rd);
    PARHourly= rd.photosyntheticallyActiveRadiationHourly;
}

```

```

        return PARHourly;
    }

    [WebMethod]
    public double[] PARHourlyBeam(double globalSolarRadiation, double[] solarElevation,
        double radiationDiffuseSky,
        double supitVanDerGroot_fractionGSRad)
    {
        //return matrix
        double[] PARHourlyBeam = new double[24];
        //instance of data-type
        RadData rd = new RadData();
        //instance of the GSRad class containing public methods
        IGSRad gs = new CRA.clima.gsrad.GSRad();
        //assign parameters
        CRA.clima.gsrad.PARHourlyBeamDiffuse.supitVanDerGroot_fractionGSRad =
            supitVanDerGroot_fractionGSRad;

        //assign inputs
        rd.globalSolarRadiation = globalSolarRadiation;
        rd.solarElevation = solarElevation;
        rd.radiationDiffuseSky = radiationDiffuseSky;
        //call for extraterrestrial radiation
        gs.PhotosyntheticallyActiveRadiation(ref rd);
        PARHourlyBeam = rd.parBeamHourly;
        return PARHourlyBeam;
    }

    [WebMethod]
    public double[] PARHourlyDiffuse(double globalSolarRadiation, double[] solarElevation,
        double radiationDiffuseSky,
        double supitVanDerGroot_fractionGSRad)
    {
        //return matrix
        double[] PARHourlyDiffuse = new double[24];
        //instance of data-type
        RadData rd = new RadData();
        //instance of the GSRad class containing public methods
        IGSRad gs = new CRA.clima.gsrad.GSRad();
        //assign parameters
        CRA.clima.gsrad.PARHourlyBeamDiffuse.supitVanDerGroot_fractionGSRad =
            supitVanDerGroot_fractionGSRad;

        //assign inputs
        rd.globalSolarRadiation = globalSolarRadiation;
        rd.solarElevation = solarElevation;
    }

```

```
rd.radiationDiffuseSky = radiationDiffuseSky;
//call for extraterrestrial radiation
gs.PhotosyntheticallyActiveRadiation(ref rd);
PARHourlyDiffuse = rd.parDiffuseHourly;
return PARHourlyDiffuse;
}
#endregion
}
}
```