Computing Exact Closed-Form Distance Distributions in Regular Polygons

CONTRIBUTION: This notebook implements the algorithm in [1] to compute: (i) the exact closed-form probability density function and (ii) the exact closed-form cumulative density function of the distance between a randomly located node and any arbitrary reference point inside a regular L-sided polygon.

These results can be used to obtain the closed-form probability density function of the Euclidean distance between any arbitrary reference point and its nth neighbor node when N nodes are uniformly and independently distributed (i.e. according to a uniform Binomial Point Process) inside a regular L-sided polygon.

CITING THIS WORK: If you use this notebook to generate distance distribution results, please cite our paper:

[1] Z. Khalid and S. Durrani, “Distance Distributions in Regular Polygons,” to appear in IEEE Transactions on Vehicular Technology, 2013.
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&tp=&arnumber=6415342
http://arxiv.org/abs/1207.5857

USAGE INSTRUCTIONS: Please note the following:
(i) Notebook assumes that the reference point is located inside the polygon.
(ii) Generally, enter reference point x and y coordinates in decimals (e.g., 0.5) , rather than as fractions (e.g. 1/2).
(iii) Sections 1-6 in the notebook generally execuate in a few seconds. Section 7 in the notebook is the time consuming part, depending upon the value of N.
(iv) The distance distributions are piecewise function. The notebook outputs values in a text file, which can be imported in Matlab for plotting.
(v) This notebook has been tested using *Mathematica* 8.0.

For comments, suggestions and bug reports, please email: salman.durrani@anu.edu.au.

Applied Signal Processing (ASP) research group,
Research School of Engineering,
The Australian National University, Canberra, Australia, May 2013.


```
ClearAll["Global`*"];
starttime = SessionTime[];
SetDirectory[NotebookDirectory[]];
```

1. Define input parameters

Define the number of sides and the circum-radius of the regular convex polygon, which is centered at (0,0)

```
L = 4;
R = 1;
```

Define the reference point, which must be located inside the regular convex polygon

```
x = 1 / 2;
y = -1 / 2;
ref = {x, y};
```

Define the tolerance to account for finite numerical precision in distance calculations

```
tol = 1 * 10^-10;
```

2. Characterise the polygon geometry: Section II in [1]

Find the polygon in-radius, side length, area, interior angle and the central angle: Equations (1) and (2) in [1]

```
inradius = R Cos[π/L];
sidelength = 2 R Sin[π/L];
area = 1/2 L R^2 Sin[2 π/L]
Θ = π (L - 2)/L;
ϑ = 2 π/L;
```

```
2
```

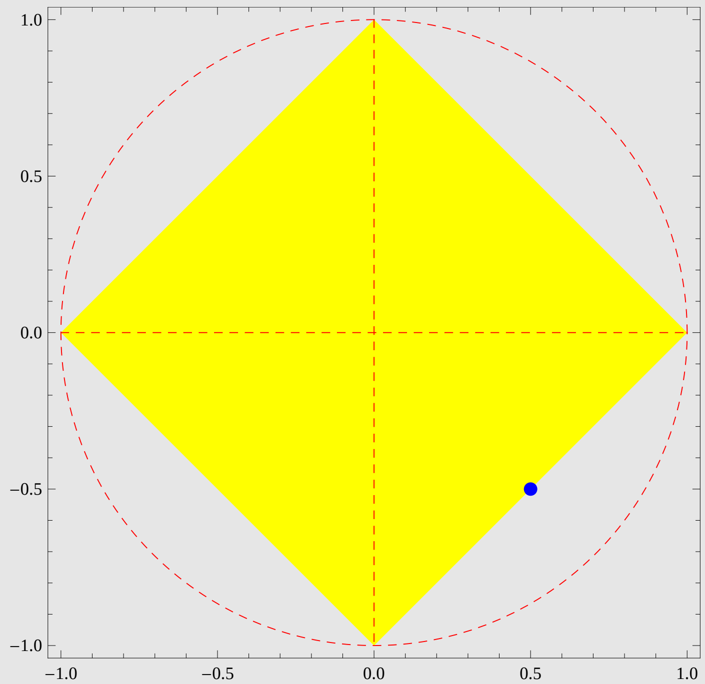
Find the coordinates of the polygon vertices; Loop from 0 to establish the first vertext at (R,0)

```
polygonvertices = {};
For[l = 0, l < L, l++, AppendTo[polygonvertices, {R Cos[l ϑ], R Sin[l ϑ]}];]
polygonvertices
```

```
{{1, 0}, {0, 1}, {-1, 0}, {0, -1}}
```

Plot the polygon and the reference point

```
Graphics[{Yellow, Polygon[polygonvertices],
  Red, Dashed, Circle[{0, 0}, R],
  Red, Dashed, Line[{{-R, 0}, {R, 0}}],
  Red, Dashed, Line[{{0, R}, {0, -R}}],
  Blue, PointSize[Large], Point[ref]}, Frame → True]
```



3. Define the distance calculation functions and then find the distances: Section II in [1]

Rotation operator: Equations (3) and (4) in [1]

```
Rotx[l_] := Cos[l ϑ] x - Sin[l ϑ] y;
Roty[l_] := Sin[l ϑ] x + Cos[l ϑ] y;
```

Distance to the vertices: Equations (5) and (6) in [1]

```
duV1[x_, y_] := Sqrt[(x - R)^2 + y^2];
duVL[x_, y_, l_] := duV1[Rotx[-(l - 1)], Roty[-(l - 1)]];
```

$$\text{duV1}[x\_, y\_] := \sqrt{(x - R)^2 + y^2};$$

Perpendicular distance to the sides: Equations (7) and (9) in [1]

$$\text{pduS1}[x\_, y\_] := \frac{\text{Abs}\left[y + x\,\text{Tan}\left[\frac{\theta}{2}\right] - R\,\text{Tan}\left[\frac{\theta}{2}\right]\right]}{\sqrt{1 + \text{Tan}\left[\frac{\theta}{2}\right]^2}};$$

```
pduSL[x_, y_, l_] := pduS1[Rotx[-(l - 1)], Roty[-(l - 1)]];
```

Shortest distance to the sides: Equations (8) and (9) in [1]

$$w[x\_, y\_] := \left\{R - \frac{(x - R)\,(\text{Cos}[\vartheta] - 1) + y\,\text{Sin}[\vartheta]}{2}, \frac{\text{Sin}[\vartheta]}{2\,(1 - \text{Cos}[\vartheta])}\,((x - R)\,(\text{Cos}[\vartheta] - 1) + y\,\text{Sin}[\vartheta])\right\};$$

```
dwV1[x_, y_] := EuclideanDistance[w[x, y], polygonvertices[[1]]];
dwV2[x_, y_] := EuclideanDistance[w[x, y], polygonvertices[[2]]];
duV1new[x_, y_] := EuclideanDistance[{x, y}, polygonvertices[[1]]];
duV2new[x_, y_] := EuclideanDistance[{x, y}, polygonvertices[[2]]];
duS1[x_, y_] := If[Max[dwV1[x, y], dwV2[x, y]] > sidelength,
   Min[duV1new[x, y], duV2new[x, y]], pduS1[x, y]];
duSL[x_, y_, l_] := duS1[Rotx[-(l - 1)], Roty[-(l - 1)]];
```

Generate the distances

```
distancetovertices = Table[duVL[x, y, l], {l, 1, L, 1}];
pdistancetosides = Table[pduSL[x, y, l], {l, 1, L, 1}];
distancetosides = Table[duSL[x, y, l], {l, 1, L, 1}];
```

Check for effects of finite precision, which can show up in the distances

```
totaldistance = Join[distancetovertices, pdistancetosides , distancetosides];
totaldistance1 = Union[totaldistance]
temp0 = Differences[totaldistance1]
```

$$\left\{0, \frac{1}{\sqrt{2}}, \sqrt{2}, \sqrt{\frac{5}{2}}\right\}$$

$$\left\{\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} + \sqrt{2}, -\sqrt{2} + \sqrt{\frac{5}{2}}\right\}$$

If present, remove the effects of finite precision in the distance calculations:

(1st check) if the absolute value of any distance (distance to a vertex or perpendicular distance to a side or minimum distance to a side) is less than a tolerance value, it is set to 0

```
distancetovertices = Chop[distancetovertices, tol];
pdistancetosides = Chop[pdistancetosides , tol];
distancetosides = Chop[distancetosides, tol];
```

(2nd check) if any two distances in the “totaldistance” list are the same, within the tolerance limit, then they are set as the same (first) distance value

```
kk = 1;
samedistance = Table[0 * l, {l, 1, Length[Select[Abs[temp0], # < tol &]], 1}]
For[l = 1, l ≤ Length[temp0], l++,
 Which[Abs[temp0[[l]]] < tol, samedistance[[kk]] = totaldistance1[[l]]];
 Which[Abs[temp0[[l]]] < tol, kk = kk + 1];]
samedistance
For[ll = 1, ll ≤ Length[samedistance], ll++,
 For[l = 1, l ≤ Length[distancetovertices], l++,
  Which[Abs[distancetovertices[[l]] - samedistance[[ll]]] < tol,
   distancetovertices[[l]] = samedistance[[ll]];];
  Which[Abs[pdistancetosides[[l]] - samedistance[[ll]]] < tol,
   pdistancetosides[[l]] = samedistance[[ll]];];
  Which[Abs[distancetosides[[l]] - samedistance[[ll]]] < tol,
   distancetosides[[l]] = samedistance[[ll]];];
 ]
]
```

```
{}
```

```
{}
```

Display the distances

```
distancetovertices
pdistancetosides
distancetosides
```

$$\left\{\frac{1}{\sqrt{2}}, \sqrt{\frac{5}{2}}, \sqrt{\frac{5}{2}}, \frac{1}{\sqrt{2}}\right\}$$

$$\left\{\frac{1}{\sqrt{2}}, \sqrt{2}, \frac{1}{\sqrt{2}}, 0\right\}$$

$$\left\{\frac{1}{\sqrt{2}}, \sqrt{2}, \frac{1}{\sqrt{2}}, 0\right\}$$

4. Border and corner effects: Section IV in [1]

Define the rule for picking a side or vertex: Follows the convention in [1] that the sides/vertices are numbered in anticlockwise direction

```
newMod[a_, b_] := If[Mod[a, b] == 0, L, Mod[a, b]];
```

Define the border and corner effects to calculate the PDF f_R(r) and store in a list: Equations (22) and (23) in [1]

```
PartialBL[l_] := 2 r ArcCos[pdistancetosides[[newMod[l, L]]] / r];
PartialCL[l_] := r (ArcCos[pdistancetosides[[newMod[l, L]]] / r] +
     ArcCos[pdistancetosides[[newMod[l - 1, L]]] / r] + π (L - 2) / L - π);
temp1 = Table[PartialBL[l], {l, 1, L, 1}];
temp2 = Table[PartialCL[l], {l, 1, L, 1}];
boundaryeffects = Chop[Flatten[Append[temp1, -temp2]], tol];
```

Define the border and corner effects to calculate the CDF F_R (r) and store in a list: Equations (13), (14), (15) and (16) in [1]

```
BL[l_] :=
  If[distancetosides[[newMod[l, L]]] ≠ 0, r^2 ArcCos[pdistancetosides[[newMod[l, L]]] / r] -
     distancetosides[[newMod[l, L]]]^2 ArcCos[pdistancetosides[[newMod[l, L]]] / distancetosides[[l]]] -
     pdistancetosides[[newMod[l, L]]] (Sqrt[r^2 - pdistancetosides[[newMod[l, L]]]^2] -
        Sqrt[(distancetosides[[newMod[l, L]]]^2 - pdistancetosides[[newMod[l, L]]]^2)]),
    r^2 ArcCos[pdistancetosides[[newMod[l, L]]] / r] - pdistancetosides[[newMod[l, L]]]
     (Sqrt[r^2 - pdistancetosides[[newMod[l, L]]]^2] -
        Sqrt[(distancetosides[[newMod[l, L]]]^2 - pdistancetosides[[newMod[l, L]]]^2)])];
CL[l_] := If[distancetovertices[[newMod[l, L]]] ≠ 0,
```

$$\frac{r^2}{2}\left(\text{ArcCos}\left[\frac{\text{pdistancetosides}[[\text{newMod}[l, L]]]}{r}\right] + \text{ArcCos}\left[\frac{\text{pdistancetosides}[[\text{newMod}[l-1, L]]]}{r}\right]\right) - \frac{\text{distancetovertices}[[\text{newMod}[l, L]]]^2}{2}\left(\text{ArcCos}\left[\frac{\text{pdistancetosides}[[\text{newMod}[l, L]]]}{\text{distancetovertices}[[\text{newMod}[l, L]]]}\right] + \text{ArcCos}\left[\frac{\text{pdistancetosides}[[\text{newMod}[l-1, L]]]}{\text{distancetovertices}[[\text{newMod}[l, L]]]}\right]\right) + \frac{\text{pdistancetosides}[[\text{newMod}[l, L]]]}{2}\left(\sqrt{(\text{distancetovertices}[[\text{newMod}[l, L]]]^2 - \text{pdistancetosides}[[\text{newMod}[l, L]]]^2)} - \sqrt{r^2 - \text{pdistancetosides}[[\text{newMod}[l, L]]]^2}\right) + \frac{\text{pdistancetosides}[[\text{newMod}[l-1, L]]]}{2}\left(\sqrt{(\text{distancetovertices}[[\text{newMod}[l, L]]]^2 - \text{pdistancetosides}[[\text{newMod}[l-1, L]]]^2)} - \sqrt{r^2 - \text{pdistancetosides}[[\text{newMod}[l-1, L]]]^2}\right) - \frac{\pi}{L}\left(r^2 - \text{distancetovertices}[[\text{newMod}[l, L]]]^2\right),$$

$$\frac{r^2}{2}\left(\text{ArcCos}\left[\frac{\text{pdistancetosides}[[\text{newMod}[l, L]]]}{r}\right] + \text{ArcCos}\left[\frac{\text{pdistancetosides}[[\text{newMod}[l-1, L]]]}{r}\right]\right) + \frac{\text{pdistancetosides}[[\text{newMod}[l, L]]]}{2}\left(\sqrt{(\text{distancetovertices}[[\text{newMod}[l, L]]]^2 - \text{pdistancetosides}[[\text{newMod}[l, L]]]^2)} - \sqrt{r^2 - \text{pdistancetosides}[[\text{newMod}[l, L]]]^2}\right) + \frac{\text{pdistancetosides}[[\text{newMod}[l-1, L]]]}{2}\left(\sqrt{(\text{distancetovertices}[[\text{newMod}[l, L]]]^2 - \text{pdistancetosides}[[\text{newMod}[l-1, L]]]^2)} -\right.$$

```
            √(r^2 - pdistancetosides[[newMod[l - 1, L]]]^2)) -
      π/L (r^2 - distancetovertices[[newMod[l, L]]]^2)];
temp1CDF = Table[BL[l], {l, 1, L, 1}];
temp2CDF = Table[CL[l], {l, 1, L, 1}];
boundaryeffectsCDF = Chop[Flatten[Append[temp1CDF, -temp2CDF]], tol];
```

5. Implement the proposed algorithm in Section V in [1] to automatically pick the correct border and corner effects for all distance ranges

Find and sort the distance vector and then find the indices k: Equation (10) and Steps 1 & 2 in Algorithm 1 in [1]

```
distances = Join[distancetosides, distancetovertices];
Sorteddistances = Sort[distances, Less]
k = Reverse[Ordering[distances, All, Greater]]
```

$$\left\{0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \sqrt{2}, \sqrt{\frac{5}{2}}, \sqrt{\frac{5}{2}}\right\}$$

```
{4, 1, 3, 5, 8, 2, 6, 7}
```

Find the correct indices of the correct border and corner effects: Step 3 in Algorithm 1 in [1]

```
temp3 = Chop[N[Differences[Sorteddistances]], tol];
indices = Flatten[Drop[Map[First, ArrayRules[temp3]], -1]]
```

```
{1, 5, 6}
```

Find the terms in the expression for the PDF f_R(r)

```
boundaryeffectsordered = boundaryeffects[[k]];
temp4 = FullSimplify[Accumulate[boundaryeffectsordered]];
answer = Apart[FullSimplify[2 π r - temp4[[indices]]]];
```

Find the terms in the expression for the CDF F_R(r)

```
boundaryeffectsorderedCDF = boundaryeffectsCDF[[k]];
temp4CDF = FullSimplify[Accumulate[boundaryeffectsorderedCDF]];
answerCDF = Apart[FullSimplify[π r^2 - temp4CDF[[indices]]]];
```

Find the unique ranges and account for the effects of finite precision

```
Uniquedistances =
  Join[Sorteddistances [[indices]], {Sorteddistances[[Length[Sorteddistances]]]}];
```

Account for the case that there may be a range with no border effects

```
Uniquedistancesnew =
 If[Uniquedistances[[1]] ≠ 0, Prepend[Uniquedistances, 0], Uniquedistances]
answernew = If[Uniquedistances[[1]] ≠ 0, Prepend[answer, 2 π r], answer]
answernewCDF = If[Uniquedistances[[1]] ≠ 0, Prepend[answerCDF, π r^2], answerCDF]
```

$$\left\{0, \frac{1}{\sqrt{2}}, \sqrt{2}, \sqrt{\frac{5}{2}}\right\}$$

$$\left\{\pi\, r, 2\, r\, \text{ArcCsc}\left[\sqrt{2}\, r\right], 2\, r\, \text{ArcCsc}\left[\sqrt{2}\, r\right] - 2\, r\, \text{ArcSec}\left[\frac{r}{\sqrt{2}}\right]\right\}$$

$$\left\{\frac{\pi\, r^2}{2}, \frac{1}{2}\sqrt{-1+2\, r^2} + r^2\, \text{ArcCsc}\left[\sqrt{2}\, r\right],\right.$$
$$\left.\sqrt{2}\sqrt{-2+r^2} + \frac{1}{2}\sqrt{-1+2\, r^2} + r^2\left(\text{ArcCsc}\left[\sqrt{2}\, r\right] - \text{ArcSec}\left[\frac{r}{\sqrt{2}}\right]\right)\right\}$$

6. Display the PDF f_R(r) and the CDF F_R(r)

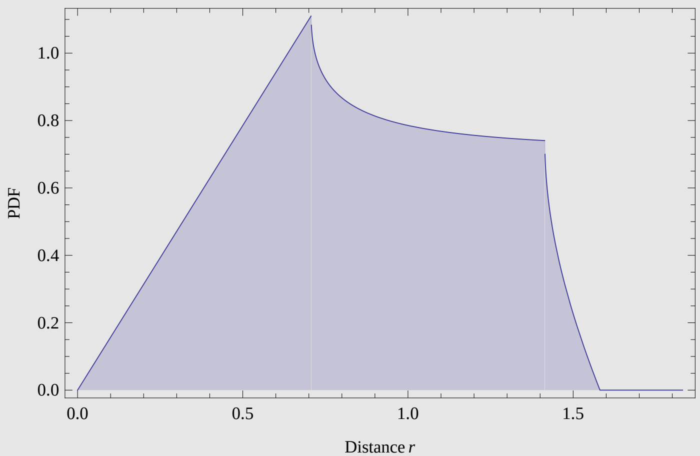
Show and plot the PDF f_R(r)

```
fRr = 1/area Piecewise[{#1, #2[[1]] ≤ r ≤ #2[[2]]} &@@@
    Transpose@{answernew, Partition[Uniquedistancesnew, 2, 1]}]
```

$$\frac{1}{2}\left(\begin{cases} \pi r & 0 \leq r \leq \frac{1}{\sqrt{2}} \\ 2 r \,\mathrm{ArcCsc}\left[\sqrt{2}\, r\right] & \frac{1}{\sqrt{2}} \leq r \leq \sqrt{2} \\ 2 r \,\mathrm{ArcCsc}\left[\sqrt{2}\, r\right] - 2 r \,\mathrm{ArcSec}\left[\frac{r}{\sqrt{2}}\right] & \sqrt{2} \leq r \leq \sqrt{\frac{5}{2}} \\ 0 & \mathrm{True} \end{cases}\right)$$

Define range and plot

```
maxrange = Max[Uniquedistances + 0.25];
Plot[fRr, {r, 0, maxrange}, Filling → Bottom, Frame → True, FrameLabel → {Distance r, PDF}]
```

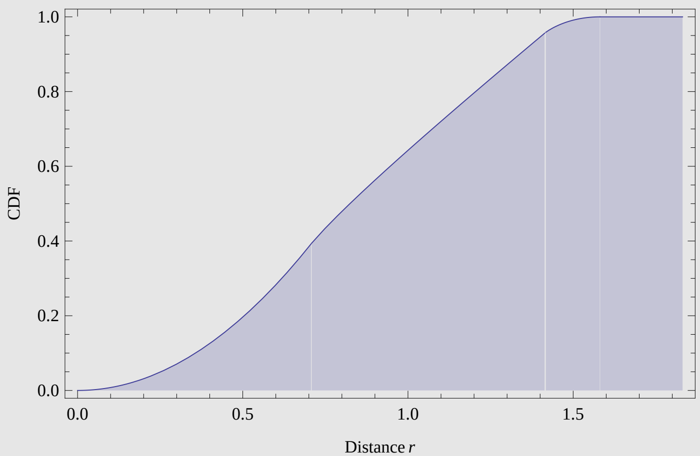


Show and plot the CDF F_R (r)

```
FRr = 1/area Piecewise[{#1, #2[[1]] ≤ r ≤ #2[[2]]} & @@@
    Transpose@{answernewCDF, Partition[Uniquedistancesnew, 2, 1]}, area]
```

$$\frac{1}{2}\left(\begin{cases} \frac{\pi r^2}{2} & 0 \le r \le \frac{1}{\sqrt{2}} \\ \frac{1}{2}\sqrt{-1+2r^2} + r^2\,\mathrm{ArcCsc}\left[\sqrt{2}\,r\right] & \frac{1}{\sqrt{2}} \le r \le \sqrt{2} \\ \sqrt{2}\sqrt{-2+r^2} + \frac{1}{2}\sqrt{-1+2r^2} + r^2\left(\mathrm{ArcCsc}\left[\sqrt{2}\,r\right] - \mathrm{ArcSec}\left[\frac{r}{\sqrt{2}}\right]\right) & \sqrt{2} \le r \le \sqrt{\frac{5}{2}} \\ 2 & \mathrm{True} \end{cases}\right)$$

```
Plot[FRr, {r, 0, maxrange}, Filling → Bottom, Frame → True, FrameLabel → {Distance r, CDF}]
```



Export the PDF and CDF values to a text file for importing and plotting in Matlab

```
ansOUT = {};
stepsize = 1/1000;
For[rr = 0.001, rr ≤ Max[Uniquedistances + 0.25],
  rr += stepsize, AppendTo[ansOUT, {r, fRr, FRr}]];
Export["result.txt", ansOUT, "table"]
```

```
result.txt
```

7. Application: n-th neighbour PDF results

Define the number of nodes

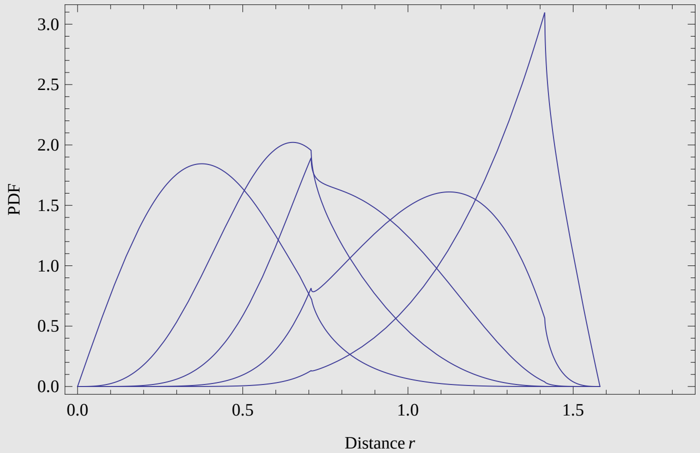
```
N = 5;
```

Define the nth neighbour PDF: Equation (12) in [1] (see reference 1 in [1])

$$\texttt{fr[N\_, n\_, r\_]} := \frac{(1 - \texttt{FRr})^{N-n}\,(\texttt{FRr})^{n-1}}{\texttt{Beta}[N - n + 1, n]}\,\texttt{fRr};$$

Plotting

```
Plot[Table[fr[N, n, r], {n , Range[1, N]}], {r, 0, maxrange},
 PlotRange → Full, Frame → True, FrameLabel → {Distance r, PDF}]
```



Display total execuation time for the notebook in seconds

```
endtime = SessionTime[];
Executiontime = (endtime - starttime)
```

```
25.6093750
```