

# **50 Berechnungsarten für Photovoltaik-Angebote & Wirtschaftlichkeitsanalysen**

Dieses Nachschlagewerk enthält die wichtigsten und modernsten PV-Berechnungsarten für Angebote, Finanzierungsmodelle, Eigenverbrauch, Speicher, CO<sub>2</sub>, Wirtschaftlichkeit und mehr.

Jede Berechnungsart ist mit Erklärung, Formel und Python-Code aufgeführt.

# Inhaltsverzeichnis

1. Jahresenergieertrag (kWh/a)
2. Stromgestehungskosten (LCOE)
3. Eigenverbrauchsquote (%)
4. Autarkiegrad (%)
5. Amortisationsdauer (Jahre)
6. Jährliche Stromkostensparnis (€)
7. Einspeisevergütung (€ p.a.)
8. Gesamtertrag über Laufzeit (kWh, €)
9. CO<sub>2</sub>-Einsparung (kg/t pro Jahr/Laufzeit)
10. Effektiver Strompreis durch PV (ct/kWh)
11. Nettobarwert (NPV)
12. Interner Zinsfuß (IRR)
13. Kapitalwert Alternativanlage
14. Kumulierte Einsparungen (€)
15. Speichergrad/Deckungsgrad Speicher (%)
16. Jährliche Degradation (%)
17. Simulation Strompreissteigerung
18. Berechnung Dachflächennutzung
19. Break-Even-Analyse
20. Szenarienvergleich (mit/ohne Speicher, etc.)
21. Performance Ratio (PR)
22. Spezifischer Ertrag (kWh/kWp)
23. Flächenspezifischer Ertrag (kWh/m<sup>2</sup>)
24. Effizienz der PV-Module (%)
25. Verschattungsverlust (%)
26. DC/AC-Überdimensionierungsfaktor
27. Temperaturkorrektur des PV-Ertrags
28. Degradationsertrag nach x Jahren
29. Gesamtkosten Wartung/Instandhaltung
30. Eigenverbrauch durch Speichererhöhung (%)
31. Optimale Speichergröße (kWh)
32. Lastverschiebepotenzial durch Verbraucher (kWh)
33. PV-Deckungsgrad für Wärmepumpe
34. Eigenkapitalrendite (ROE)
35. Kapitaldienstfähigkeitsprüfung
36. Restwert der Anlage (nach x Jahren)
37. Steuerliche Abschreibung (linear, degressiv)
38. Ersparnis durch Fördermittel/Kredite
39. Netzanschlusskosten
40. Vergleich PV vs. Balkonkraftwerk
41. Ertragsverlust durch Wechselrichter-Degradation
42. Notstromfähigkeit (kWh/Tag)
43. Batteriezyklus-Kalkulation
44. Ladeprofil-Simulation für E-Auto
45. Kumulierte CO<sub>2</sub>-Einsparung (über x Jahre)
46. Inflationsausgleich in der Wirtschaftlichkeitsrechnung
47. Investitionsszenarien (A/B/C)
48. Anlagenerweiterung (Ertrag, Kosten, Amortisation)
49. Ausfallwahrscheinlichkeit/Kosten Risikoanalyse
50. Peak-Shaving-Effekt

## 1. Jahresenergieertrag (kWh/a)

**Erklärung:** Jährlicher Stromertrag der PV-Anlage.

**Formel:**

$\text{Jahresertrag} = \text{Anlagenleistung [kWp]} \times \text{spezifischer Ertrag [kWh/kWp]}$

**Python-Code:**

```
def jahresertrag_kw(kWp, spezifischer_ertrag): return kWp * spezifischer_ertrag
```

## 2. Stromgestehungskosten (LCOE)

**Erklärung:** Kosten pro erzeugter kWh über die gesamte Lebensdauer.

**Formel:**

$\text{LCOE} = \text{Summe aller Kosten} / \text{Summe aller erzeugten kWh}$

**Python-Code:**

```
def lcoe(gesamt_kosten, summe_kwh): return gesamt_kosten / summe_kwh
```

## 3. Eigenverbrauchsquote (%)

**Erklärung:** Anteil des PV-Stroms, der selbst verbraucht wird.

**Formel:**

$\text{Eigenverbrauchsquote} = \text{Eigenverbrauch} / \text{Gesamtproduktion} \times 100$

**Python-Code:**

```
def eigenverbrauchsquote(eigenverbrauch_kwh, gesamtertrag_kwh): return  
eigenverbrauch_kwh / gesamtertrag_kwh * 100
```

## 4. Autarkiegrad (%)

**Erklärung:** Wie viel Prozent des Strombedarfs durch die PV-Anlage gedeckt werden.

**Formel:**

$\text{Autarkiegrad} = \text{Eigenverbrauch} / \text{Gesamtverbrauch} \times 100$

**Python-Code:**

```
def autarkiegrad(eigenverbrauch_kwh, gesamtverbrauch_kwh): return  
eigenverbrauch_kwh / gesamtverbrauch_kwh * 100
```

## 5. Amortisationsdauer (Jahre)

**Erklärung:** Zeit bis zur vollständigen Kostendeckung (Break-even).

**Formel:**

$\text{Amortisation} = \text{Investitionskosten} / \text{jährliche Einsparung}$

**Python-Code:**

```
def amortisationsdauer(invest, jaehrliche_ersparnis): return invest /  
jaehrliche_ersparnis
```

## 6. Jährliche Stromkostensparnis (€)

**Erklärung:** Geldersparnis pro Jahr durch Eigenverbrauch.

**Formel:**

$\text{Ersparnis} = \text{Eigenverbrauch} \times \text{Strompreis}$

**Python-Code:**

```
def stromkostenersparnis(eigenverbrauch_kwh, strompreis): return
eigenverbrauch_kwh * strompreis
```

## 7. Einspeisevergütung (€ p.a.)

**Erklärung:** Einnahmen durch Stromeinspeisung ins Netz.

**Formel:**

$\text{Vergütung} = \text{eingespeiste\_kWh} \times \text{Einspeisevergütungssatz}$

**Python-Code:**

```
def einspeiseverguetung(einspeisung_kwh, satz_euro): return einspeisung_kwh *
satz_euro
```

## 8. Gesamtertrag über Laufzeit (kWh, €)

**Erklärung:** Stromertrag oder Einnahmen über z. B. 20 Jahre.

**Formel:**

$\text{Gesamtertrag} = \text{Jahresertrag} \times \text{Laufzeit}$

**Python-Code:**

```
def gesamtertrag_laufzeit(jahresertrag, laufzeit_jahre): return jahresertrag *
laufzeit_jahre
```

## 9. CO<sub>2</sub>-Einsparung (kg/t pro Jahr/Laufzeit)

**Erklärung:** Vermeidung von CO<sub>2</sub>-Ausstoß durch PV-Strom.

**Formel:**

$\text{CO}_2\text{-Ersparnis} = \text{Jahresertrag} \times \text{Strommix-Faktor (z.B. 0.4 kg/kWh)}$

**Python-Code:**

```
def co2_ersparnis(jahresertrag_kwh, co2_faktor=0.4): return jahresertrag_kwh *
co2_faktor
```

## 10. Effektiver Strompreis durch PV (ct/kWh)

**Erklärung:** Effektive Kosten je selbst erzeugter Kilowattstunde.

**Formel:**

$\text{PV-Strompreis} = \text{Gesamtkosten} / \text{Gesamtertrag} \times 100 \text{ (für Cent)}$

**Python-Code:**

```
def effektiver_pv_strompreis(gesamt_kosten, gesamtertrag_kwh): return
gesamt_kosten / gesamtertrag_kwh * 100
```

## 11. Nettobarwert (NPV)

**Erklärung:** Heutiger Wert aller künftigen Ein- und Auszahlungen.

**Formel:**

$\text{NPV} = \text{Summe (Einnahmen - Ausgaben)} / (1 + \text{Zinssatz})^{**} \text{Jahr}$

**Python-Code:**

```
def npv(cashflows, zinssatz): return sum([cf / (1+zinssatz)**i for i, cf in
enumerate(cashflows)])
```

## 12. Interner Zinsfuß (IRR)

**Erklärung:** Effektive jährliche Rendite der Investition.

**Formel:**

$IRR = \text{Zinssatz, bei dem Kapitalwert} = 0.$

**Python-Code:**

```
import numpy as np def irr(cashflows): return np.irr(cashflows)
```

## 13. Kapitalwert Alternativanlage

**Erklärung:** Vergleich mit alternativen Investments.

**Formel:**

$\text{Endkapital} = \text{Invest} \times (1 + \text{Zinssatz})^{**} \text{Jahre}$

**Python-Code:**

```
def alternativanlage_kapitalwert(invest, zinssatz, laufzeit): return invest * (1 + zinssatz)**laufzeit
```

## 14. Kumulierte Einsparungen (€)

**Erklärung:** Gesamte Stromkostensparnisse über Laufzeit.

**Formel:**

$\text{Kumulierte Ersparnis} = \text{jährliche Ersparnis} \times \text{Laufzeit}$

**Python-Code:**

```
def kumulierte_ersparnis(jaehrliche_ersparnis, laufzeit): return jaehrliche_ersparnis * laufzeit
```

## 15. Speichergrad/Deckungsgrad Speicher (%)

**Erklärung:** Anteil des Eigenverbrauchs, der vom Speicher abgedeckt wird.

**Formel:**

$\text{Speichergrad} = \text{gespeicherter Eigenverbrauch} / \text{gesamter Eigenverbrauch} \times 100$

**Python-Code:**

```
def speichergrad(gespeichert, eigenverbrauch): return gespeichert / eigenverbrauch * 100
```

## 16. Jährliche Degradation (%)

**Erklärung:** Leistungsverlust der Module pro Jahr.

**Formel:**

$\text{Leistung\_nach\_n\_Jahren} = \text{Startleistung} \times (1 - \text{Degradation})^{**} n$

**Python-Code:**

```
def leistung_nach_jahren(startleistung, degradation, jahre): return startleistung * ((1 - degradation) ** jahre)
```

## 17. Simulation Strompreissteigerung

**Erklärung:** Wie sich die Stromkosten mit der Zeit entwickeln.

**Formel:**

$\text{Stromkosten\_nach\_n\_Jahren} = \text{Anfangskosten} \times (1 + \text{Steigerung})^{**} n$

**Python-Code:**

```
def stromkosten_nach_jahren(anfangskosten, steigerung, jahre): return anfangskosten * ((1 + steigerung) ** jahre)
```

## 18. Berechnung Dachflächennutzung

**Erklärung:** Wie viele Module passen aufs Dach?

**Formel:**

Anzahl Module = (verfügbare Dachfläche) / (Modulfläche)

**Python-Code:**

```
def module_auf_dach(dachflaeche_m2, modul_l, modul_b): modulflaeche = modul_l * modul_b return int(dachflaeche_m2 / modulflaeche)
```

## 19. Break-Even-Analyse

**Erklärung:** Wann werden die Investitionskosten durch Einsparungen gedeckt?

**Formel:**

Break-even = Invest / jährliche Ersparnis

**Python-Code:**

```
def break_even_jahr(invest, jaehrliche_ersparnis): return int(invest // jaehrliche_ersparnis) + 1
```

## 20. Szenarienvergleich (mit/ohne Speicher, etc.)

**Erklärung:** Vergleich unterschiedlicher Anlagenauslegungen.

**Formel:**

Vergleich aller KPIs je Szenario.

**Python-Code:**

```
def szenarienvergleich(configs): results = [] for config in configs: result = {} results.append(result) return results
```

## 21. Performance Ratio (PR)

**Erklärung:** Maß für die technische Betriebsqualität der Anlage.

**Formel:**

PR = tatsächlicher Ertrag / (Globalstrahlung × installierte Leistung)

**Python-Code:**

```
def performance_ratio(ertrag_kwh, globalstrahlung_kwh, kWp): return ertrag_kwh / (globalstrahlung_kwh * kWp)
```

## 22. Spezifischer Ertrag (kWh/kWp)

**Erklärung:** Ertrag je installiertem kWp.

**Formel:**

spezifischer Ertrag = Jahresertrag / kWp

**Python-Code:**

```
def spezifischer_ertrag(jahresertrag_kwh, kWp): return jahresertrag_kwh / kWp
```

## 23. Flächenspezifischer Ertrag (kWh/m²)

**Erklärung:** Ertrag pro Quadratmeter belegter Dachfläche.

**Formel:**

$\text{Ertrag}_{m2} = \text{Jahresertrag} / \text{belegte Fläche}$

**Python-Code:**

```
def ertrag_je_flaeche(jahresertrag_kwh, belegte_flaeche_m2): return
jahresertrag_kwh / belegte_flaeche_m2
```

## 24. Effizienz der PV-Module (%)

**Erklärung:** Wirkungsgrad der Module bezogen auf Fläche.

**Formel:**

$\text{Effizienz} = \text{Modulleistung} / (\text{Fläche} \times 1000)$

**Python-Code:**

```
def pv_effizienz(modulleistung_watt, modulflaeche_m2): return modulleistung_watt /
(modulflaeche_m2 * 1000) * 100
```

## 25. Verschattungsverlust (%)

**Erklärung:** Verlust durch zeitweise Verschattung.

**Formel:**

$\text{Verschattungsverlust} = 1 - (\text{Ertrag verschattet} / \text{Ertrag optimal})$

**Python-Code:**

```
def verschattungsverlust(ertrag_verschattet, ertrag_optimal): return 1 -
(ertrag_verschattet / ertrag_optimal)
```

## 26. DC/AC-Überdimensionierungsfaktor

**Erklärung:** Verhältnis PV-Modulleistung zu Wechselrichterleistung.

**Formel:**

$\text{Überdimensionierung} = \text{PV-Leistung} / \text{WR-Leistung}$

**Python-Code:**

```
def dc_ac_ueberdimensionierung(pv_leistung_kwp, wr_leistung_kw): return
pv_leistung_kwp / wr_leistung_kw
```

## 27. Temperaturkorrektur des PV-Ertrags

**Erklärung:** Ertrag bei abweichender Modultemperatur.

**Formel:**

$P_{\text{real}} = P_{\text{nom}} * [1 + TK * (T - T_{\text{ref}})]$

**Python-Code:**

```
def pv_leistung_temp_korr(p_nom, tk_percent, temp, t_ref=25): return p_nom * (1 +
tk_percent/100 * (temp - t_ref))
```

## 28. Degradationsertrag nach x Jahren

**Erklärung:** Prognose der Ertragsminderung durch Alterung.

**Formel:**

$\text{Leistung} = \text{Anfangswert} \times (1 - \text{jährliche Deg.}) ** \text{Jahre}$

**Python-Code:**

```
def degradations_ertrag(startwert, jahres_deg, jahre): return startwert * ((1 -
jahres_deg) ** jahre)
```

## 29. Gesamtkosten Wartung/Instandhaltung

**Erklärung:** Summe aller Wartungskosten über die Laufzeit.

**Formel:**

$\text{Wartungskosten}_{\text{total}} = \text{jährl. Wartung} \times \text{Laufzeit}$

**Python-Code:**

```
def wartungskosten_total(jahrliche_wartung, laufzeit): return jahrliche_wartung * laufzeit
```

## 30. Eigenverbrauch durch Speichererhöhung (%)

**Erklärung:** Steigerung des Eigenverbrauchs durch Batteriespeicher.

**Formel:**

$\text{Eigenverbrauch}_{\text{neu}} = \text{Funktion}(\text{Speichergröße}, \text{Lastprofil}, \text{PV-Profil})$

**Python-Code:**

```
def eigenverbrauch_mit_speicher(alt, zusatz_speicher_prozent): return alt + zusatz_speicher_prozent
```

## 31. Optimale Speichergröße (kWh)

**Erklärung:** Empfohlene Größe für maximalen Eigenverbrauch.

**Formel:**

$\text{optimale Speichergröße} \approx \text{Tagesverbrauch} \times (1 - \text{Verluste})$

**Python-Code:**

```
def optimale_speichergröße(tagesverbrauch, pv_profil, verluste=0.1): return tagesverbrauch * (1 - verluste)
```

## 32. Lastverschiebepotenzial durch Verbraucher (kWh)

**Erklärung:** Potenzial zur Nutzung von PV-Strom durch verschiebbare Verbraucher.

**Formel:**

$\text{Lastverschiebung} = \text{Summe steuerbare Verbraucher} \times \text{PV-Überschuss}$

**Python-Code:**

```
def lastverschiebepotenzial(steuerbare_kwh, pv_ueberschuss_kwh): return min(steuerbare_kwh, pv_ueberschuss_kwh)
```

## 33. PV-Deckungsgrad für Wärmepumpe

**Erklärung:** Anteil des WP-Verbrauchs durch PV gedeckt.

**Formel:**

$\text{PV-Deckung}_{\text{WP}} = \text{PV-Überschuss} / \text{WP-Jahresverbrauch} \times 100$

**Python-Code:**

```
def pv_deckung_wp(pv_ueberschuss, wp_jahresverbrauch): return pv_ueberschuss / wp_jahresverbrauch * 100
```

## 34. Eigenkapitalrendite (ROE)

**Erklärung:** Rendite auf das eingesetzte Eigenkapital.



**Formel:**

$ROE = (\text{Gewinn} / \text{eingesetztes EK}) \times 100$

**Python-Code:**

```
def roe(gewinn, eigenkapital): return (gewinn / eigenkapital) * 100
```

## 35. Kapitaldienstfähigkeitsprüfung

**Erklärung:** Kann der Kunde die Finanzierung leisten?

**Formel:**

$\text{Kapitaldienst} = \text{Jahresüberschuss} / \text{Annuität}$

**Python-Code:**

```
def kapitaldienstfaehigkeit(jahresueberschuss, annuitaet): return  
jahresueberschuss / annuitaet
```

## 36. Restwert der Anlage (nach x Jahren)

**Erklärung:** Wert der PV-Anlage nach der Abschreibungsdauer.

**Formel:**

$\text{Restwert} = \text{Invest} \times (1 - \text{jährliche Abschreibung})^{**} \text{Jahre}$

**Python-Code:**

```
def restwert(invest, abschreibung, jahre): return invest * ((1 - abschreibung) **  
jahre)
```

## 37. Steuerliche Abschreibung (linear, degressiv)

**Erklärung:** Vorteil durch steuerliche AfA.

**Formel:**

$\text{AfA linear} = \text{Investition} / \text{Abschreibungsdauer}$

**Python-Code:**

```
def afa_linear(invest, abschreibungsjahre): return invest / abschreibungsjahre
```

## 38. Ersparnis durch Fördermittel/Kredite

**Erklärung:** Abzug der Fördersumme, z. B. KfW, BAFA.

**Formel:**

$\text{Effektive Kosten} = \text{Invest} - \text{Förderung}$

**Python-Code:**

```
def foerderersparnis(invest, foerderung): return invest - foerderung
```

## 39. Netzanschlusskosten

**Erklärung:** Zusatzkosten für Zählertausch, Anmeldung etc.

**Formel:**

$\text{Netzanschlusskosten} = \text{Summe aller Anschlusskosten}$

**Python-Code:**

```
def netzanschlusskosten(*kosten): return sum(kosten)
```

## 40. Vergleich PV vs. Balkonkraftwerk

**Erklärung:** Kostenvorteil und Ertrag im Vergleich.

**Formel:**

Vergleich: Kosten- und Ertragsdifferenz beider Systeme

**Python-Code:**

```
def vergleiche_pv_bkw(kosten_pv, ertrag_pv, kosten_bkw, ertrag_bkw): return {
    'kosten_diff': kosten_pv - kosten_bkw, 'ertrag_diff': ertrag_pv - ertrag_bkw }
```

## 41. Ertragsverlust durch Wechselrichter-Degradation

**Erklärung:** Leistungsverlust des Wechselrichters mit der Zeit.

**Formel:**

$\text{Ertrag\_nach\_WR-Verlust} = \text{Ertrag} \times (1 - \text{WR\_Degradationsrate})^{**} \text{Jahre}$

**Python-Code:**

```
def wr_degradation_ertrag(ertrag_start, degradationsrate, jahre): return
ertrag_start * ((1 - degradationsrate) ** jahre)
```

## 42. Notstromfähigkeit (kWh/Tag)

**Erklärung:** Wie viel Energie steht bei Stromausfall zur Verfügung?

**Formel:**

$\text{Notstromfähigkeit} = \text{nutzbare Speichergröße} \times \text{Effizienz}$

**Python-Code:**

```
def notstrom_kapazitaet(speicher_kwh, nutzungsgrad): return speicher_kwh *
nutzungsgrad
```

## 43. Batteriezyklus-Kalkulation

**Erklärung:** Wie viele Ladezyklen hat der Speicher?

**Formel:**

$\text{Lebensdauer} = \text{max. Zyklen} / \text{Zyklen/Jahr}$

**Python-Code:**

```
def batterie_lebensdauer(max_zyklen, zyklen_pro_jahr): return max_zyklen /
zyklen_pro_jahr
```

## 44. Ladeprofil-Simulation für E-Auto

**Erklärung:** Wie viel PV-Strom kann das E-Auto nutzen?

**Formel:**

$\text{Geladene Energiemenge} = \min(\text{PV-Überschuss}, \text{Bedarf}) \times \text{Effizienz}$

**Python-Code:**

```
def eauto_ladeprofil(pv_ueberschuss, bedarf, ladeeffizienz=0.9): return
min(pv_ueberschuss, bedarf) * ladeeffizienz
```

## 45. Kumulierte CO<sub>2</sub>-Einsparung (über x Jahre)

**Erklärung:** CO<sub>2</sub>-Einsparung über die gesamte Laufzeit.

**Formel:**

$\text{CO}_2\text{\_ges} = \text{jährliche Ersparnis} \times \text{Laufzeit}$

**Python-Code:**

```
def co2_kumuliert(jahres_ersparnis, laufzeit): return jahres_ersparnis * laufzeit
```

## 46. Inflationsausgleich in der Wirtschaftlichkeitsrechnung

**Erklärung:** Auswirkung der Inflation auf Erträge und Kosten.

**Formel:**

$$\text{Wert\_nach\_Inflation} = \text{Wert} / (1 + \text{Inflationsrate})^{**} \text{Jahre}$$

**Python-Code:**

```
def wert_nach_inflation(wert, inflationsrate, jahre): return wert / ((1 + inflationsrate) ** jahre)
```

## 47. Investitionsszenarien (A/B/C)

**Erklärung:** Vergleich unterschiedlicher Anlagen- und Finanzierungsmodelle.

**Formel:**

Vergleich: alle KPIs je Szenario

**Python-Code:**

```
def invest_szenarien(vergleiche): return {name: kpis for name, kpis in vergleiche.items()}
```

## 48. Anlagenerweiterung (Ertrag, Kosten, Amortisation)

**Erklärung:** Kalkulation nach Nachrüstung (Module/Speicher).

**Formel:**

Neuberechnung aller KPIs nach Erweiterung

**Python-Code:**

```
def anlagenerweiterung(alt, neu): return {k: neu[k] - alt.get(k, 0) for k in neu}
```

## 49. Ausfallwahrscheinlichkeit/Kosten Risikoanalyse

**Erklärung:** Risikoanalyse von Ausfällen und Reparaturen.

**Formel:**

$$\text{Risiko} = \text{Schadenshöhe} \times \text{Ausfallwahrscheinlichkeit}$$

**Python-Code:**

```
def risikoanalyse(schadenshoehe, wahrscheinlichkeit): return schadenshoehe * wahrscheinlichkeit
```

## 50. Peak-Shaving-Effekt

**Erklärung:** Reduzierung von Lastspitzen durch PV und Speicher.

**Formel:**

$$\text{Peak shaving} = \text{max. Last} - \text{gemessene Last nach Optimierung}$$

**Python-Code:**

```
def peak_shaving(max_last, opt_last): return max_last - opt_last
```

## Notizen